



Université  
de Liège

Université de Liège

Faculté des sciences appliquées

**Travail de fin d'études réalisé en vue de l'obtention du grade de master Ingénieur Civil en informatique par Quentin Smetz**

Année académique 2013-2014

Encadrement académique : PhD Marc Van Droogenbroeck

Encadrement industriel : PhD Sébastien Jodogne

Membres supplémentaires du jury : PhD Pierre Geurts, PhD Christophe Phillips

Développement d'un logiciel léger et portable  
d'images médicales 3D, selon les standards  
médicaux et sur base d'Orthanc

---

Quentin Smetz

# Résumé

Ce travail de fin d'études porte sur le **Développement d'un logiciel léger et portable d'images médicales 3D, selon les standards médicaux et sur base d'Orthanc**. Le travail explore la conception étape par étape d'une interface graphique permettant la visualisation paramétrée de modalités d'images 2D et 3D ainsi que la fusion de celles-ci. Il s'inscrit dans un développement plus large effectué au CHU de Liège : Orthanc, serveur DICOM complet.

La conception présente les objectifs et une série de diagrammes UML, après une phase d'étude des attentes du milieu médical. L'implémentation repose sur les bibliothèques Qt et VTK, ainsi que sur celle développée au CHU de Liège. Des précisions sont apportées sur les standards médicaux qui sont pris en compte lors de l'implémentation.

Le résultat final est un logiciel complet permettant une utilisation ergonomique des fonctionnalités principales d'un logiciel d'imagerie médicale. Le code produit est quant à lui conçu pour évoluer comme cela est décrit tout au long de l'implémentation. Ce code est destiné à une distribution open-source.

Quentin Smetz

Faculté des sciences appliquées

Université de Liège

2013-2014

# Remerciements

J'aimerais particulièrement remercier l'encadrement extérieur qu'a assuré tout au long de l'année monsieur Sébastien Jodogne, développeur d'`Orthanc`. Le résultat final n'aurait pu être atteint sans sa réactivité et son implication. Je remercie aussi le docteur Philippe Martinive et monsieur Éric Lenaerts qui m'ont accordé un peu de leur temps pour répondre à mes questions ciblant les médecins et les physiciens médicaux. Enfin, je remercie mon promoteur, monsieur Van Droogenbroeck, grâce à qui ce travail a pu être initié. Dans le but de n'oublier personne, je remercie aussi toutes les personnes s'étant intéressées à mon travail.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Contenu du travail . . . . .	6
1.2	Démarche suivie . . . . .	6
1.3	Sources . . . . .	7
<b>2</b>	<b>Imagerie médicale et DICOM</b>	<b>8</b>
2.1	Introduction . . . . .	8
2.2	Format de fichier . . . . .	8
2.3	Orthanc . . . . .	9
<b>3</b>	<b>Analyse des besoins</b>	<b>11</b>
3.1	Introduction . . . . .	11
3.2	Étude des attentes du milieu médical . . . . .	12
3.3	Résumé . . . . .	12
<b>4</b>	<b>Choix technologiques</b>	<b>14</b>
4.1	Qt . . . . .	14
4.2	VTK . . . . .	14
<b>5</b>	<b>Architecture logicielle</b>	<b>16</b>
5.1	Cycle de développement préliminaire . . . . .	16
5.2	Chargement de fichier DICOM . . . . .	17
5.3	Visualisation de modalités d'imagerie 3D . . . . .	18
5.4	Application de transformation d'image . . . . .	19
5.5	Modification du layout de visualisation . . . . .	24
5.6	Fusion des séries . . . . .	25
5.7	Ajouts supplémentaires . . . . .	26
5.8	Organisation selon une architecture MVC . . . . .	27
<b>6</b>	<b>Résultats</b>	<b>29</b>
6.1	Lancement du logiciel . . . . .	29
6.2	Chargement d'une série DICOM . . . . .	30

6.3 Application de transformation d'image . . . . .	32
6.4 Modification du layout de visualisation . . . . .	37
6.5 Fusion des séries . . . . .	39
<b>7 Améliorations futures</b>	<b>42</b>
7.1 Visualisation d'autres modalités 3D . . . . .	42
7.2 Mesures . . . . .	42
7.3 Annotations . . . . .	42
7.4 Alignement automatique des séries . . . . .	43
7.5 Autres . . . . .	43
<b>8 Conclusions</b>	<b>44</b>
<b>A Formulaire d'enquête</b>	<b>46</b>
A.1 Formulaire . . . . .	46
A.2 Résultats . . . . .	49

# Chapitre 1

## Introduction

### 1.1 Contenu du travail

Ce travail de fin d'études porte sur la réalisation d'un logiciel d'imagerie médicale. On peut distinguer plusieurs compétences différentes devant être mises en œuvre pour atteindre les objectifs finaux :

**Ingénierie logicielle** Le produit final est un logiciel relativement conséquent. Il comporte une interface utilisateur (GUI) intégrant une visualisation d'images 2D et 3D récupérées depuis le réseau, lesquelles pouvant être transformées et fusionnées. Toutes ces tâches doivent donc s'harmoniser autour d'une implémentation rigoureuse.

**Ergonomie** Le produit final est destiné à une utilisation médicale. L'interface doit répondre aux besoins du personnel hospitalier l'utilisant.

**Intégration** Au-delà de la réalisation elle-même, le logiciel doit s'intégrer dans le cadre d'un développement plus large, `Orthanc`. Dès lors, l'implémentation se voudra la plus modulaire possible et la documentation doit permettre une évolution future par d'autres développeurs.

### 1.2 Démarche suivie

Afin de réaliser efficacement le logiciel, la démarche initiale ci-dessous a été suivie :

1. Apprentissage de la librairie `Qt` afin d'avoir une vision d'ensemble sur ce qui est facilement réalisable ou non
2. Préparation d'un cahier des charges
3. Tests de logiciels existants en imagerie médicale afin de mieux cerner les points forts à intégrer et les erreurs à éviter
4. Réalisation d'une étude (questionnaire) envoyée au milieu hospitalier pour mieux comprendre leurs besoins

## 5. Rencontre au CHU avec un médecin et un physicien médical

À partir de là, le travail se compose d'un mélange de suivi en milieu hospitalier, d'implémentation du logiciel et de documentation de celui-ci, selon les standards de modèles UML.

### 1.3 Sources

Ce travail de fin d'études étant essentiellement un travail de réalisation, relativement peu de références apparaîtront dans la bibliographie. Les documentations des outils utilisés seront toutefois citées. Bien entendu, celles-ci ont été complétées par de nombreuses informations, découvertes sur base de discussions trouvées sur le net. Les spécifications techniques de certains standards médicaux sont aussi précisées.

# Chapitre 2

## Imagerie médicale et DICOM

### 2.1 Introduction

L'imagerie médicale se décompose en toute une série de « modalités ». Ces différents types d'imagerie médicale sont, entre autres, les suivants :

**CT** (Computed Tomography), un faisceau de rayons X balaye le corps, tranche par tranche. Des mesures de l'absorption de ces rayons tout autour de la tranche permettent la construction d'une image de celle-ci. À partir des images des tranches, le volume 3D peut être reconstruit. Un CT scan permet une visualisation anatomique du corps.

**PET** (Positron Emission Tomography), imagerie obtenue par mesure des émissions produites par un composé radioactif injecté au préalable dans le corps. Une image PET permet de visualiser l'activité du corps.

**MRI** (Magnetic Resonance Imaging) obtenue par exploitation de l'énergie fournie par les atomes du corps après leur excitation à l'aide de champs magnétiques. Ce type d'imagerie et ses variantes permet aussi bien une visualisation anatomique que fonctionnelle du corps.

La liste ci-dessus est loin d'être exhaustive, de nombreuses autres modalités d'image sont manipulées en milieu hospitalier, comme l'échographie pour fournir un autre exemple.

La difficulté s'annonçant dans la conception d'un logiciel d'imagerie médicale est dès lors la diversité des images à traiter. Cependant, cette difficulté sera contournée par l'utilisation du format de fichier **DICOM** qui permettra, à peu de chose près, de ne pas tenir compte du type d'image utilisé.

### 2.2 Format de fichier

Le format de fichier **DICOM** est un standard pour l'imagerie médicale. L'acronyme **DICOM** signifie : Digital Imaging and Communications in Medicine. Il permet notamment de classer des images médicales selon une hiérarchie à quatre niveaux :

- Les patients

- Les études
- Les séries
- Les instances

Ainsi, un fichier DICOM contient une image 2D (l'**instance**) d'une coupe d'un certain volume 3D (la **série**) obtenu lors d'une **étude** à laquelle s'est soumis un certain **patient**.

Toutes les informations « textuelles » relatives à ces différents niveaux sont donc aussi contenues dans un fichier DICOM. Celui-ci est organisé selon un modèle XML, sous la forme d'une série de balises avec un nom (un *tag*) et un contenu. Le standard contient plusieurs milliers de tags différents ; fort heureusement un fichier DICOM n'en utilise jamais autant à la fois et seule une poignée d'entre eux sera nécessaire dans le cadre de ce travail.

## 2.3 Orthanc

Orthanc est l'infrastructure actuellement développée au CHU à partir de laquelle est né ce travail de fin d'études. Ce chapitre vise à décrire sa philosophie et ses fonctionnalités. La plupart des textes de cette section seront des traductions légèrement retouchées des descriptions que l'on peut trouver à son sujet sur le site officiel (en date du mois d'août 2014) :

<http://www.orthanc-server.com>

### 2.3.1 Philosophie

Orthanc vise à fournir un serveur de fichiers DICOM autonome, simple, mais puissant. Il est conçu pour améliorer les flux de fichiers DICOM dans les hôpitaux et à soutenir la recherche sur l'analyse automatique des images médicales.

Orthanc peut transformer n'importe quel ordinateur fonctionnant sous Windows ou Linux en une bibliothèque de fichiers DICOM (en d'autres termes, un système de mini-PACS). Son architecture est légère et autonome, ce qui signifie qu'aucune administration de base de données complexe n'est nécessaire, ni l'installation de dépendances extérieures.

Orthanc est conçu pour répondre aux objectifs suivants :

- Faciliter le traitement de fichiers DICOM pour la routine clinique (par exemple C-Move, C-Store et C-Find SCU).
- Faciliter la gestion des données pour la routine clinique et la recherche médicale (mini-PACS).
- Mettre à disposition des images DICOM à la communauté scientifique (pour faciliter l'analyse automatisée des images médicales).

Pour atteindre ces objectifs, `Orthanc` est :

- Léger et rapide (écrit en C++)
- Autonome (toutes les dépendances peuvent être liées statiquement)
- Multiplate-forme (au moins Linux et Windows)
- Conforme à la norme DICOM
- « Programmer-friendly » (API REST, JSON, PNG)

### 2.3.2 API REST

Ce qui rend `Orthanc` unique est qu'il fournit une API REST. Grâce à cette fonction importante, il est possible de commander `Orthanc` à l'aide de n'importe quel langage informatique. Les balises DICOM des images médicales stockées peuvent être téléchargées dans le format de fichier JSON. En outre, les images PNG standard peuvent être générées à la volée par `Orthanc` à partir des instances des fichiers DICOM.

`Orthanc` permet ainsi à ses utilisateurs de se concentrer sur le contenu des fichiers DICOM, en ignorant la complexité du format et du protocole DICOM.

# Chapitre 3

## Analyse des besoins

### 3.1 Introduction

Un cahier des charges préliminaire a été mis au point. Une seconde partie de l'analyse a permis de laisser de côté les parties inutiles, non essentielles à ce travail ou nécessitant une évolution d'`Orthanc`. Celles-ci seront démarquées par leur police italique dans la liste ci-dessous.

- Charger des fichiers DICOM dans le logiciel (*réseau ou système de fichier*)
- Navigation dans `Orthanc`
- Lecture de modalités d'imagerie 3D
- Fusion d'images (entre instances du même patient ou non)
- Fenêtrage (Hounsfield)
- Possibilité d'appliquer des « colormap »
- Support du MIP (Maximum Intensity Projection)
- Changement de layout d'affichage des séries
- Flexibilité pour la taille des vues (et plein écran)
- Flexibilité sur le nombre de fenêtres ouvertes
- Application de transformations géométriques aux vues
- *Support du RT-STRUCT*
- *Support du 2D+t et du 3D+t*
- *Indépendance ou dépendance entre différentes vues*
- *Intégration du travail d'alignement automatique créé par un doctorant*
- *Prise de mesure sur les vues*
- *Découpage d'un volume 3D selon un angle de coupe*
- *Annotation des fichiers DICOM*

## 3.2 Étude des attentes du milieu médical

### 3.2.1 Rencontre avec les spécialistes

Une rencontre avec des spécialistes utilisant des visionneuses 3D dans le cadre de leur travail à l'hôpital a été organisée. Celle-ci a permis de mieux cerner les attentes des deux principaux secteurs utilisant ce genre de logiciel, à savoir les radiothérapeutes et les physiciens médicaux.

Le dialogue a aussi permis la réalisation d'un formulaire de sondage plus adéquat (voir section suivante).

### 3.2.2 Sondage

Un formulaire a été envoyé au milieu hospitalier afin de mieux cerner leurs besoins au niveau de l'interface et des fonctionnalités ainsi que pour obtenir tout autre information pouvant être utile à la réalisation d'un logiciel adapté à l'utilisation dont il fera l'objet. Ce formulaire ainsi que les divers graphiques d'analyse des réponses qu'il a engendrés sont présents en annexe A.1 et A.2, page 46 et 49.

En résumé, de ce sondage, il est ressorti les résultats suivants :

- Les médecins s'attendent à utiliser toute forme de fonctionnalités. En effet, peu de fonctionnalités proposées se sont révélées inutiles.
- Le logiciel doit à tout prix être ergonomique. C'est en effet le manque principal des logiciels utilisés actuellement.

### 3.2.3 Conclusion

On remarque une certaine divergence parmi les avis sur ce que doit contenir un bon logiciel d'imagerie médicale. En effet, celui-ci devant être utilisé par une série de personnes aux profils différents, le logiciel devrait être capable de s'adapter en fonction des besoins de tous. La conclusion est, qu'en fait, le logiciel doit être capable de supporter quasi toutes les fonctionnalités de base proposées car peu semblent superflues. Cependant, l'identification des relations entre les différents types de besoin, comme exposée ci-après, permet de découper le travail en différentes étapes.

## 3.3 Résumé

### UML

L'ensemble du cahier des charges préliminaire est décrit par un diagramme de cas d'utilisation. Ceux-ci sont regroupés en cycles de développement permettant ainsi de segmenter le travail en différentes étapes.

Le diagramme complet est représenté sur la figure 3.1, les cycles de développement étant encadrés à l'aide de couleurs différentes.



FIGURE 3.1 – Diagramme de cas d'utilisation

# Chapitre 4

## Choix technologiques

Le logiciel sera codé en **C++**, notamment parce que c'est dans ce langage qu'**Orthanc** est implémenté. L'interface graphique sera configurée à l'aide des bibliothèques **Qt** et **VTK**.

### 4.1 Qt

**Qt** est une interface de programmation orientée objet, codée en **C++**, permettant notamment de développer des applications fenêtrées. Cette bibliothèque, ou plutôt cet ensemble de bibliothèques, a été choisie dans le cadre de ce TFE pour ses larges possibilités, son excellente documentation et, surtout, son aspect portable. En effet, bien que l'implémentation et les tests du logiciel ont été effectués sous Linux, le but est de pouvoir facilement le rendre compatible sous d'autres systèmes d'exploitation. **Qt** assurera donc principalement la gestion du design du logiciel et de son interface. D'autres possibilités sont offertes par la bibliothèque et ne manqueront pas non plus d'être utilisées. On peut citer notamment le système de signaux et de slots permettant d'ajouter aux classes **C++** :

**Des signaux** Ce sont des événements pouvant être déclenchés et interceptés.

**Des slots** Ce sont des méthodes particulières qui peuvent être reliées à un signal, de sorte que lorsque ce dernier survient, la méthode en question est appelée.

La version de **Qt** choisie est la version **4.7**. La dernière version en date (5) étant sortie depuis moins d'un an lors du lancement de ce travail de fin d'études, elle n'a pas été considérée suffisamment en place et supportée par la communauté informatique pour être utilisée. Cependant, le portage vers la version 5 d'un logiciel codé pour la version **4.7** est assez rapide pour effectuer ce changement si nécessaire pour d'éventuelles améliorations futures.

### 4.2 VTK

**VTK** (Visualization ToolKit) est une bibliothèque open-source spécialisée dans la visualisation et le traitement de données 2D et 3D. Elle constitue donc le choix parfait pour l'application dont

traite ce TFE. À l'instar de **Qt**, la bibliothèque est vaste et ses possibilités le sont donc aussi. Son implémentation repose sur **OpenGL**, bibliothèque connue et stable pour le traitement de données 2D et 3D.

De plus, la bibliothèque **VTK** est bien adaptée pour être intégrée à une interface codée à l'aide de **Qt**. En effet, la classe **QVTKWidget** permet de former un lien transparent entre les deux bibliothèques.

# Chapitre 5

## Architecture logicielle

Cette partie s'intéressera à l'implémentation et à la documentation des fonctionnalités du logiciel. Le diagramme de cas d'utilisation présenté précédemment (voir figure 3.1) sera utilisé pour implémenter le logiciel dans l'ordre défini des cycles de développement. Notons que seules les parties implémentées lors de ce travail de fin d'études seront exposées dans cette section.

### 5.1 Cycle de développement préliminaire

Avant d'entamer le développement des composantes décrites par les cas d'utilisation, une étape préliminaire regroupant certains points semble indispensable.

#### 5.1.1 Personnalisation de l'interface

Beaucoup d'éléments du développement sont des éléments graphiques (fenêtres et widgets). Il semble intéressant de ne pas utiliser les éléments graphiques de **Qt** tels quels mais de construire des éléments personnalisés de ceux-ci. Un mécanisme d'héritage est alors mis en œuvre afin de pouvoir utiliser ces éléments de la même façon que ceux de base fournis par **Qt** mais permettant la mise en place d'un design spécifique au logiciel. Notons aussi que des éléments hors design peuvent y être insérés, comme de nouveaux signaux et slots pour faciliter le développement. Ainsi, pour chaque élément graphique propre à **Qt**, une nouvelle classe fille est créée et toutes ces nouvelles classes sont regroupées dans un espace de nom à part entière nommé **customwidget**.

#### 5.1.2 Fenêtre principale

Tout d'abord, la fenêtre principale du programme peut être vue comme un élément à part du diagramme de cas d'utilisation présenté précédemment. En effet, tous les éléments qui y sont présentés s'articulent autour de celle-ci.

Cette fenêtre sert en fait de lien entre les cas d'utilisation de chargement de série DICOM et d'exploitation de celle-ci.

Cette fenêtre doit aussi pouvoir être visible par beaucoup d'éléments du programme, par exemple pour afficher un message sur la barre de statut. Dès lors, la classe de la fenêtre principale, `ViewerWindow` a été implémentée selon un modèle singleton, la rendant de fait unique et accessible.

## 5.2 Chargement de fichier DICOM

Pour charger les fichiers depuis `Orthanc`, plusieurs éléments ont été distingués :

- Une boîte de dialogue dédiée à la demande des identifiants de connexion vers `Orthanc` et maintenant cette connexion. La classe associée est nommée `OrthancConnectionDialog`.
- Une seconde boîte de dialogue gérant les résultats de la première et permettant la navigation à travers les éléments d'`Orthanc`. La classe dédiée à cette tâche est nommée `OrthancDialog`.
- Des « threads » (fils d'exécution ou tâches) spécialisés dans le chargement des fichiers, depuis le réseau, à l'aide d'`Orthanc`. Ces threads sont représentés par la classe appelée `LoadSeriesThread`.

La classe `OrthancDialog` permettra donc de sélectionner une série selon une hiérarchie patient - étude - série, comme présenté dans la section 2.

De plus, il a été décidé de créer une classe `OkCancelDialog` dont hériteront toutes les classes représentant des boîtes de dialogue afin de proposer automatiquement un bouton « Valider » et un bouton « Annuler » appelant respectivement les méthodes dédiées à l'acceptation et au rejet des boîtes de dialogue Qt.

Le chargement de série DICOM se fait à l'aide de la bibliothèque C++ développée par le CHU dont il a été question au chapitre 2. Le thread dédié à cette tâche se charge de préparer le volume 3D et les informations nécessaires à sa visualisation. De toute évidence, une classe dédiée au regroupement de ces informations a donc été implémentée. Cette dernière est nommée `SeriesData`. Les informations chargées sont notamment le nom du patient, de l'étude et de la série, le type d'imagerie (CT, PET, ...) et d'autres informations qui seront abordées plus tard.

Au final, on opte donc pour le diagramme de classe présenté sur la figure 5.1.

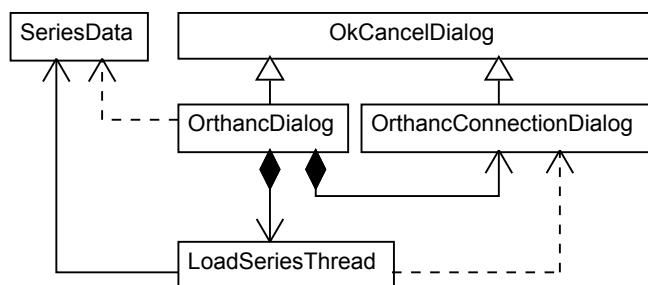


FIGURE 5.1 – Diagramme de classe pour le chargement de fichiers DICOM

## Calcul du volume 3D

Comme mentionné ci-dessus, le volume 3D est construit à l'aide de la bibliothèque développée par le CHU. Cependant, il a été remarqué que le calcul de la taille des voxels était incomplet. Les voxels sont en quelque sorte les pixels d'une image en trois dimensions. Le volume est formé à partir d'une série de petits parallélépipèdes rectangles, nommés voxels, caractérisés par une hauteur, une longueur et une largeur. Pour chaque coupe, le fichier DICOM précise la longueur et la largeur des voxels. En revanche, la hauteur doit être calculée à partir d'autres données. Le logiciel se charge donc de la calculer.

Les tags 0020-0032 et 0020-0037 (`ImagePositionPatient` et `ImageOrientationPatient`) doivent, dans ce cas, être récupérés. Le premier indique la position (en millimètre) des coordonnées  $x$ ,  $y$  et  $z$  du centre du voxel du coin supérieur gauche. Le calcul de la distance entre ces voxels supérieurs gauches pour deux tranches consécutives suffit donc à calculer la hauteur d'un voxel. Cependant, il a été choisi d'effectuer le calcul autrement. Le second tag mentionné indique les cosinus directeurs de la première ligne et de la première colonne de voxels, autrement dit les coordonnées des axes suivant la direction de cette ligne et de cette colonne. Grâce à un produit vectoriel entre ces deux vecteurs et une normalisation, on obtient un vecteur unitaire normal au plan de l'image DICOM. Le produit scalaire de ce vecteur avec celui entre les voxels supérieurs gauches de deux coupes consécutives permet alors de trouver la hauteur d'un voxel. Le résultat obtenu avec la première méthode est normalement identique. Cependant, certains fichiers DICOM apparemment imprécis fournissent une différence entre ces deux calculs, bien que invisible à l'oeil nu une fois les volumes reconstruits avec l'une ou l'autre valeur. La seconde méthode a été choisie car elle semble plus fiable. Par exemple, dans le cas hypothétique où deux coupes consécutives seraient décalées, de tailles différentes, de tailles de voxels différentes ou autres, le calcul resterait correct pour obtenir la distance entre deux coupes.

Notons que les tags 0018-0050 et 0018-0088 (`SliceThickness` et `SpacingBetweenSlices`) existent et pourraient faire penser qu'ils traitent de la hauteur des voxels. En réalité, leur signification est incertaine et ces tags ne sont pas obligés d'être présents, tandis que ceux présentés ci-dessus le sont systématiquement. Il a été remarqué que la valeur fournie par le premier de ces tags pour les images CT semblent être effectivement la hauteur des voxels. Cependant, pour d'autres modalités, sa valeur semble plutôt indiquer la hauteur de la coupe acquise et non la distance entre deux coupes (hauteur de voxel). Le second tag est rarement présent.

## 5.3 Visualisation de modalités d'imagerie 3D

Maintenant que le programme permet d'instancier des objets regroupant toutes les informations nécessaires d'une série 3D, il faut pouvoir utiliser ces informations pour la visualisation. Pour cela, on distingue divers éléments :

- Une interface dédiée à la gestion de la série chargée. La classe gérant cette partie est

appelée `SeriesInterface`.

- De petites interfaces se chargeant de gérer une vue particulière de la série chargée (volume et coupes). La classe créée pour cette fonctionnalité est la classe `SubInterface`. Celle-ci est alors spécialisée en deux autres classes propres à la gestion des coupes et des volumes, `SliceSubInterface` et `VolumeSubInterface`.
- Des éléments graphiques spécialisés dans l'affichage de la série. Ces éléments utiliseront donc la bibliothèque VTK et hériteront de la classe `QVTKWidget`, assurant un lien transparent entre les bibliothèques Qt et VTK. La classe permettant ces affichages est nommée `SeriesViewer` et est spécialisée en `SeriesSliceViewer` et `SeriesVolumeViewer`. La visualisation à l'aide de VTK fonctionne sous trois niveaux importants : la fenêtre de visualisation, les moteurs de rendus (et leurs caméras) et les éléments visualisés. Toute l'implémentation de la visualisation consiste en un paramétrage de ces éléments.

L'analyse des besoins a permis d'affirmer que l'affichage des coupes frontales, sagittales et transverses ainsi que du volume 3D était systématiquement nécessaire. Chaque instance de `SeriesInterface` contiendra donc trois instances de `SliceSubInterface` et une instance de `VolumeSubInterface`. Ces dernières contiendront une instance de `SeriesSliceViewer` et une instance de `SeriesVolumeViewer`, respectivement.

Notons que les vues sont conçues pour respecter la convention médicale pour la radiologie, c'est-à-dire :

- Afficher la gauche du corps sur la droite de l'image et vice-versa
- Les coupes transverses sont vues de dessous
- Les coupes sagittales sont vues de gauche
- Les coupes frontales sont vues de face

En neurologie, les conventions peuvent être inversées, le logiciel ne permet cependant pas cette possibilité pour le moment.

On arrive dès lors au diagramme de classe présenté sur la figure 5.2.

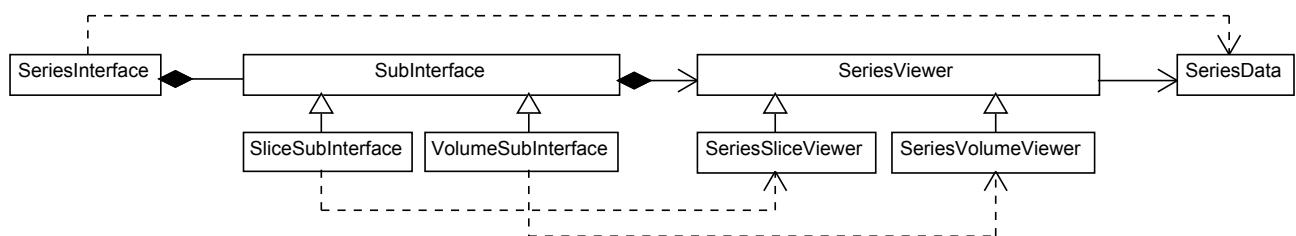


FIGURE 5.2 – Diagramme de classe pour la visualisation des séries chargées

## 5.4 Application de transformation d'image

Les fenêtres de visualisation affichent pour l'instant le volume ou les coupes à l'état « brut ». Plus précisément, chaque point des données est associé à une valeur et les visionneuses affichent

les données 3D ou les coupes en considérant que le point de plus faible valeur est noir et le point de plus haute valeur est blanc.

La transformation d'image permet de personnaliser la manière dont les données sont affichées.

### 5.4.1 Zoom et déplacement

La fonction de zoom et de translation de la vue est une fonctionnalité de base de la classe parente des fenêtres de visualisation, `QVTKWidget`.

### 5.4.2 Fenêtrage Hounsfield

Le fenêtrage Hounsfield permet de ne mettre en valeur qu'une certaine plage numérique parmi les valeurs des données. De base, la plus faible valeur est représentée en noir, et la plus haute en blanc. Grâce au fenêtrage Hounsfield, les valeurs correspondant au noir et au blanc peuvent être modifiées (une échelle linéaire de nuance de gris étant toujours utilisée entre ces deux valeurs).

#### Utilité

À certaines plages de valeurs numériques correspondent différents tissus humains. Les unités Hounsfield, UH, fournissent une échelle normalisée pour décrire la radio-densité. À titre d'exemple, la figure 5.3 indique la plage de valeurs UH pour quelques tissus humains. Notons que d'autres unités de mesures peuvent être utilisées, comme la densité optique ; le principe reste le même.

Matière	UH
Air	-1000
Poumon	-500
Graisse	-100 à -50
Eau	0
Liquide cérébro-spinal	15
Rein	30
Sang	30 à 45
Muscle	10 à 40
Matière grise	37 à 45
Matière blanche	20 à 30
Foie	40 à 60
Tissus mous	100 à 300
Os	700 à 3000

FIGURE 5.3 – Valeurs en unités Hounsfield associées à différentes matières

Le format de fichier DICOM contient des tags qui permettent de faire correspondre les valeurs associées aux points de l'image contenue à des valeurs selon une certaine unité. L'unité

vers laquelle la transformation a lieu est spécifiée par le tag 0028-1054, `RescaleType`. Par défaut, il s'agira des unités Hounsfield. Les tags 0028-1052 et 0028-1053, `RescaleIntercept` et `RescaleSlope` permettent de faire une correspondance linéaire suivant la formule :

$$\text{physicalValue} = \text{pixelValue} \times \text{RescaleSlope} + \text{RescaleIntercept}$$

Par défaut, en unité Hounsfield, les valeurs de `RescaleIntercept` et de `RescaleSlope` seront respectivement égales à  $-1024$  et  $1$ . En effet, les niveaux de gris étant généralement codés sur 12 bits (soit 4096 valeurs possibles), cela permet de couvrir la plage numérique allant de  $-1024$  à  $3071$  ce qui couvre les valeurs principales d'unités Hounsfield utilisées.

Notons aussi qu'au lieu d'une correspondance linéaire, le format de fichier DICOM permet aussi qu'une table de mise en correspondance (LUT) soit fournie à la place des tags précédemment présentés. Ce cas particulier ne sera pas traité dans ce travail.

## Implémentation

Les tags DICOM précédemment cités sont par conséquent chargés dans l'objet `SeriesData` et ce dernier permet de passer des unités « bruts » vers les unités physiques et vice-versa.

Pour permettre la customisation du fenêtrage utilisé pour afficher les niveaux de gris souhaités, de nouvelles classes sont implémentées.

D'une part, la classe `SeriesViewer` précédemment présentée doit maintenant être capable de modifier la visualisation sur base de certains paramètres. Dès lors, la classe `ViewConfiguration` est implémentée pour regrouper l'ensemble de ces paramètres. Une plage de valeurs représentant le fenêtrage (Hounsfield) désiré y est ajoutée comme paramètre à l'aide d'une nouvelle classe `Range`.

D'autre part, l'utilisateur doit pouvoir indiquer les paramètres qu'il désire. C'est pourquoi une classe `ViewConfigurationDialog` est construite afin de représenter une boîte de dialogue permettant la génération d'objets `ViewConfiguration`. Notons que pour toujours ajouter plus de flexibilité au logiciel, cette classe de boîte de dialogue et ses futures spécialisations fournissent la possibilité de retenir une précédente configuration pour y revenir plus tard à l'aide d'un simple clic sur un bouton « réinitialiser ». Pour faire varier le fenêtrage, une boîte de dialogue personnalisée sera implémentée comme nous le verrons plus loin dans ce rapport.

## Valeurs prédefinies

La norme DICOM prévoit aussi de définir un ou des fenêtrages idéaux pour la visualisation de l'image médicale contenue dans un fichier. Ces fenêtrages prédefinis sont chargés par le logiciel et l'utilisateur peut naviguer parmi ceux-ci.

En plus de ces fenêtrages « de base », le logiciel propose aussi des fenêtrages « utiles » pour la visualisation spécifique de certains tissus (poumons, muscles, ...). Notons que ces fenêtrages

personnalisés ne sont utiles que si l'unité utilisée est l'unité Hounsfield, ce qui est généralement le cas pour les CT scans.

### 5.4.3 Colormap

Les images médicales sont formées de pixels dont seule l'intensité, leur valeur, est définie. C'est pourquoi elles sont la plupart du temps connues pour être affichées en noir et blanc. Cependant, par une simple mise en correspondance entre ces valeurs d'intensité et des couleurs, ces images peuvent être affichées suivant une certaine coloration.

Les palettes de couleur peuvent être appliquées de multiples façons. Au cours de ce TFE, de nombreuses méthodes ont été testées, mais par soucis de simplicité et d'ergonomie, seules deux méthodes ont été conservées. Toutes deux se basent sur le fait que l'utilisateur puisse créer une table où chaque entrée associe à une valeur réelle comprise entre 0 et 1, une couleur. Cette table est alors appliquée de façon différente selon la méthode choisie.

**Méthode 1** Les valeurs 0 et 1 de la table sont associées aux valeurs de fenêtrage (Hounsfield) vu précédemment. Les couleurs définies dans l'intervalle sont interpolées linéairement selon leurs composantes RGB.

**Méthode 2** Les valeurs 0 et 1 de la table sont associées aux valeurs minimales et maximales que peuvent prendre l'intensité des pixels. Les couleurs ne sont pas interpolées linéairement mais maintenues constantes entre chaque changement de couleur. Le fenêtrage (Hounsfield) permettra de faire varier l'intensité de ces couleurs (composante « Value » dans le système colorimétrique HSV).

Pour la création de la palette de couleur, le choix est laissé à l'utilisateur. Il peut :

- Ajouter lui-même ses couleurs à la table en choisissant les valeurs de teinte, saturation et valeurs et en déplaçant un curseur entre les valeurs 0 et 1.
- Charger un fichier standardisé pour la création de colormap, les fichiers LUT.

### Fichiers LUT

Les fichiers LUT utilisés sont des fichiers binaires contenant 768 valeurs comprises entre 0 et 255. Les 256 premières sont les valeurs de quantité de rouge à attribuer aux 256 niveaux de gris allant linéairement du blanc au noir. Les 256 suivantes sont les valeurs de quantité de vert et les 256 dernières sont les valeurs de quantité de bleu.

Le logiciel se charge de transformer cette palette de couleur à 256 couleurs en une nouvelle où le minimum nécessaire de couleurs est utilisé, en supposant, entre celles-ci, une progression linéaire des composantes RGB de ces couleurs. Notons aussi que le logiciel est conçu pour fonctionner avec des fichiers LUT à n'importe quel nombre de couleurs (pas forcément 256).

## Implémentation

Tous ces nouveaux éléments impliquent la création de nouvelles classes. Comme la plupart du temps la customisation du fenêtrage et celle de la palette de couleur appliquée sont liées, une seule boîte de dialogue regroupe ces deux fonctionnalités, `HounsfieldColormapDialog` héritant de la classe `ViewConfigurationDialog` décrite précédemment. La classe `ViewConfiguration`, elle, se voit complétée d'un nouvel objet, instance d'une nouvelle classe nommée `Colormap`. Notons enfin que la boîte de dialogue est composée d'éléments graphiques gérés par les classes `HounsfieldWidget`, `ColormapWidget` et `ColorbarWidget`.

La figure 5.4 représente le diagramme de classe des fonctionnalités décrites dans ce chapitre. Notons que d'autres éléments sont présents sur ce diagramme et seront présentés dans une section ultérieure.

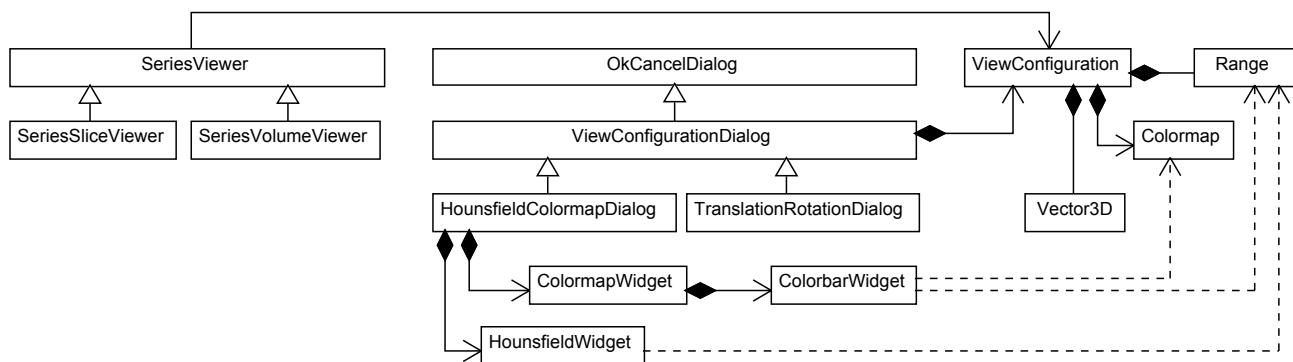


FIGURE 5.4 – Diagramme de classe pour l'application de transformations à la série chargée

### 5.4.4 Maximum Intensity Projection

Pour visualiser les volumes 3D, VTK utilise la technique de « raycasting », c'est-à-dire celle du lancer de rayons. Des propriétés sont associées au volume, notamment :

- Une fonction d'opacité attribuant une opacité de 0 à toute valeur en dessous de la valeur inférieure du fenêtrage (Hounsfield) et croissant linéairement jusqu'à l'opacité maximale, associée à toute valeur au dessus de la valeur supérieure de ce fenêtrage.
- Une fonction de coloration, tout comme pour l'affichage des coupes.
- Une fonction de lancer de rayon permettant le dessin du volume.

C'est en modifiant la fonction de lancer de rayon que la visualisation du volume peut passer du mode « normal » au mode MIP. De base, la fonction effectue une composition des valeurs d'opacité et de couleur en chaque point pour construire le volume. En mode MIP, comme son nom l'indique, le rayon se charge de faire ressortir la valeur maximale *d'opacité* qu'il trouve sur son chemin. Concrètement, sur une image CT cela fera ressortir les os (vu qu'ils ont une valeur élevée en unités hounsfield) et sur une image PET, cela fera ressortir les zones d'activités importantes du composé radioactif injecté pour acquérir l'image.

## 5.5 Modification du layout de visualisation

Une des demandes phares du milieu hospitalier était l'ergonomie du logiciel. Les logiciels actuels sont en effet en général assez complets mais leur utilisation est tout sauf intuitive.

Pour accomplir cette tâche, des changements au niveau des classes `SeriesInterface`, `SubInterface` et leurs spécialisations sont nécessaires. De plus, pour distinguer la partie gestion de la visualisation de la partie modification du layout de visualisation, une nouvelle classe `DisplayInterface` est créée comme parente de la classe `SeriesInterface`. Ceci conduit au diagramme de classe présenté sur la figure 5.5.

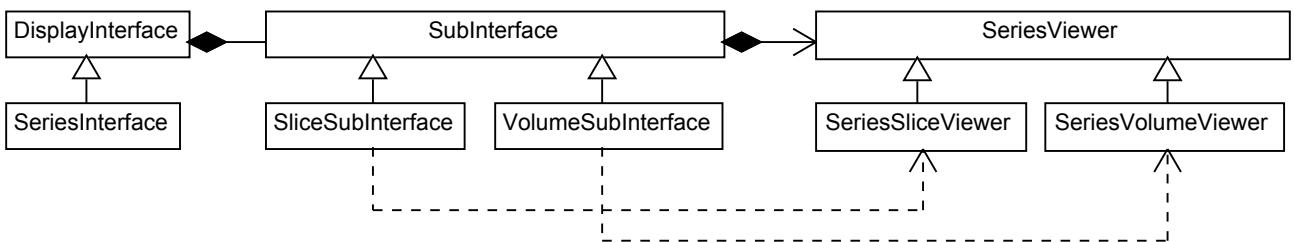


FIGURE 5.5 – Diagramme de classe pour la modification du layout de visualisation

### 5.5.1 Changement des dimensions d'une vue

À l'aide de « splitters » (diviseurs d'écran), chaque vue peut être redimensionnée et ce, de manière intuitive.

#### Plein écran

Un simple clic sur le bouton correspondant permet d'afficher toute l'interface de visualisation d'une série en plein écran.

Un double clic sur une vue permet de masquer toutes les autres pour ne plus afficher que celle-ci sur toute la surface de la fenêtre.

### 5.5.2 Modification de l'agencement des vues

La barre d'outil permet de choisir entre les modes de visualisation qui avaient été proposés au milieu hospitalier.

### 5.5.3 Modification de l'agencement des séries

Le choix est laissé à l'utilisateur pour la position de l'interface de visualisation d'une série ouverte : il peut la laisser intégrée au groupement par onglets de toutes les séries ouvertes, la détacher vers une fenêtre indépendante ou, comme déjà mentionné, la faire passer en plein écran.

## 5.6 Fusion des séries

La fusion des séries pouvait être implémentée de diverses manières. Le choix a été fait de tirer profit de toute l'architecture créée jusqu'alors pour considérer qu'une fusion est une simple collection d'éléments de base, agrémentée de méthodes pour adapter le comportement de ces éléments les uns par rapport aux autres.

De nouvelles classes gérant ces collections d'éléments de base sont ainsi créées :

**MergedSeriesInterface** héritant des propriétés de **DisplayInterface** pour la flexibilité au niveau du layout de visualisation. La classe contiendra une série de liens vers des instances de **SeriesInterface**.

**MergedSeriesViewer** regroupera, de manière similaire, un ensemble de liens vers des instances de **SeriesViewer**.

**Viewer** regroupe les éléments communs des classes **SeriesViewer** et **MergedSeriesViewer** en la nommant parente de ces dernières.

**MergedSeriesSliceViewer** spécialise la classe **MergedSeriesViewer** pour l'affichage de coupes fusionnées.

**MergedSeriesVolumeViewer** spécialise la classe **MergedSeriesViewer** pour l'affichage de volumes fusionnés.

**FusionDialog** assure une interface utilisateur pour la création de fusions.

On arrive dès lors à la hiérarchie de classe présentée sur la figure 5.6.

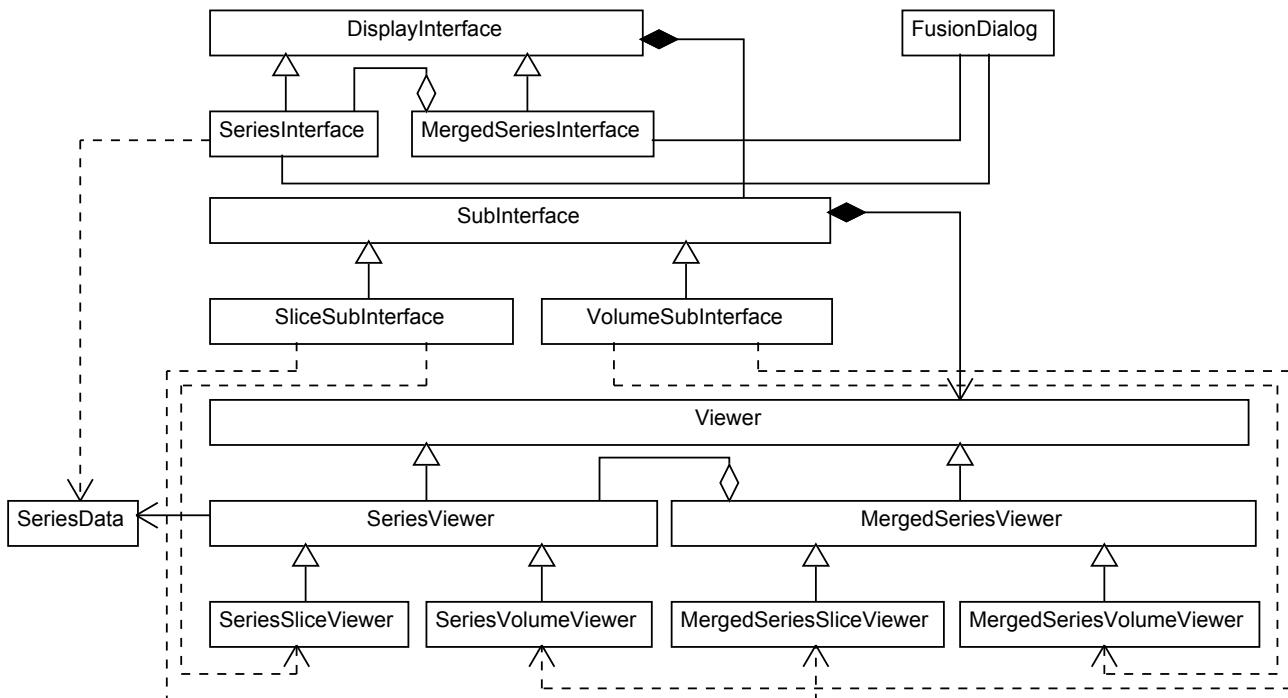


FIGURE 5.6 – Diagramme de classe pour la visualisation de séries fusionnées

### 5.6.1 Implémentation

L'implémentation tire donc profit de l'affichage dans les interfaces de base (sans fusion). Les composants de fusion se chargent de « détacher » les éléments affichés dans ces interfaces et de les empiler avec une certaine opacité pour laisser apparaître tous les éléments ainsi ajoutés. La fusion consiste donc en une superposition d'images non opaques. Les opacités des différents éléments fusionnés peuvent être contrôlées par l'utilisateur. Par défaut, les opacités sont égales à 0.5, sauf pour le premier élément fusionné où elle est fixée à 1. En effet, les éléments fusionnés seront empilés en plaçant le dernier élément ajouté « au dessus », c'est-à-dire le plus proche de la caméra visualisant la scène. Notons que cet empilement est effectué lors de chaque déplacement à travers les coupes, afin de toujours placer un écart suffisant et égal entre les éléments fusionnés (quelque soit le nombre de coupes que chaque série fusionnée contient).

### 5.6.2 Alignement

Lorsque l'on désire fusionner différentes séries, c'est en général pour visualiser des informations de différents types en même temps. Le parfait exemple consiste en la fusion d'une image CT et de son image PET correspondante. On arrive ainsi à visualiser l'activité du corps en même temps que son anatomie. Cependant, lorsque l'on superpose ces séries, celles-ci ne seront jamais parfaitement alignées. De légères translations et/ou rotations d'une image par rapport à une autre seront nécessaires. Aussi, lorsque l'on veut se déplacer à travers les coupes, un système où il n'existe qu'un bouton « passer à la coupe suivante » ne suffit plus. En effet, les séries peuvent ne pas contenir le même nombre de coupes. Dès lors, divers éléments ont été rajoutés et modifiés :

- La possibilité de déplacer et de tourner les éléments visualisés (et non plus déplacer la vue par rapport aux éléments visualisés). Pour cela, une nouvelle classe `Vector3D` est créée pour compléter les paramètres de la classe `ViewConfiguration` et une nouvelle spécialisation, `TranslationRotationDialog`, de la classe `ViewConfigurationDialog` est implémentée.
- Les sliders contrôlant le déplacement à travers les coupes (de la classe `SliceSubInterface`) permettent à présent « un déplacement en unité physique ». Ceci permet, lors d'une fusion, d'afficher les coupes les plus proches du point physique que l'on veut visualiser.

## 5.7 Ajouts supplémentaires

### Configuration du programme

Une nouvelle classe suivant le modèle du singleton a été implémentée : `ProgramConfiguration`. Celle-ci permet, à partir d'un fichier de configuration, de charger à l'exécution certaines informations utiles au programme. Ces informations sont les suivantes :

- Le chemin menant au dossier contenant les images et icônes utiles au design du logiciel
- Le chemin menant vers un dossier contenant des fichiers de palette de couleur (LUT) afin de le proposer automatiquement à l'utilisateur lorsqu'il veut charger un fichier de ce type
- Les informations utiles pour la connexion à `Orthanc` afin de les indiquer par défaut à l'utilisateur lors de la demande de connexion
- Les valeurs de plages prédéfinies à proposer pour les fenêtrages Hounsfield

## 5.8 Organisation selon une architecture MVC

Le logiciel étant destiné à des améliorations futures par des développeurs différents, il est nécessaire d'avoir une organisation rigoureuse dans le développement. Outre une documentation complète et détaillée, le code a finalement été divisé selon une architecture **MVC**, modèle - vue - contrôleur.

Bien que la frontière puisse parfois être floue entre ces différents concepts, les fichiers ont été organisés à l'intérieur d'une hiérarchie de dossiers :

**Code/Model** contient la partie « modèle ». Idéalement, ce dossier ne doit contenir aucun code d'affichage à l'aide de VTK ou Qt mais uniquement les codes de classes représentant les formats de données, ce qui est le cas.

**Code/View/Qt** contient les composants graphiques de librairie Qt personnalisés pour les besoins du logiciel ainsi que les boîtes de dialogue pour le paramétrage des vues.

**Code/View/VTK** contient tous les algorithmes d'affichage des séries à l'aide de VTK.

**Code/Controller** contient le code assurant le lien entre les parties modèles et vues. La frontière entre cette composante et la partie « vue Qt » est souvent arbitraire.

Le diagramme final du logiciel est représenté sur la figure 5.7. La division selon l'architecture MVC est elle aussi représentée. Notons toutefois que quelques liens moins importants ont été omis afin d'aérer le diagramme (par exemple, certaines classes utilisent des éléments de modèle comme Range pour des informations internes, ou encore, les SubInterface spécialisées manipulent des Viewer spécialisés comme expliqué précédemment). Aussi, la classe DoubleSlider et les autres éléments graphiques appartenant au namespace customwidget ne sont pas mentionnés. Enfin, les utilisations et spécialisations de classes provenant de Qt ou VTK ne sont bien sûr pas représentées.

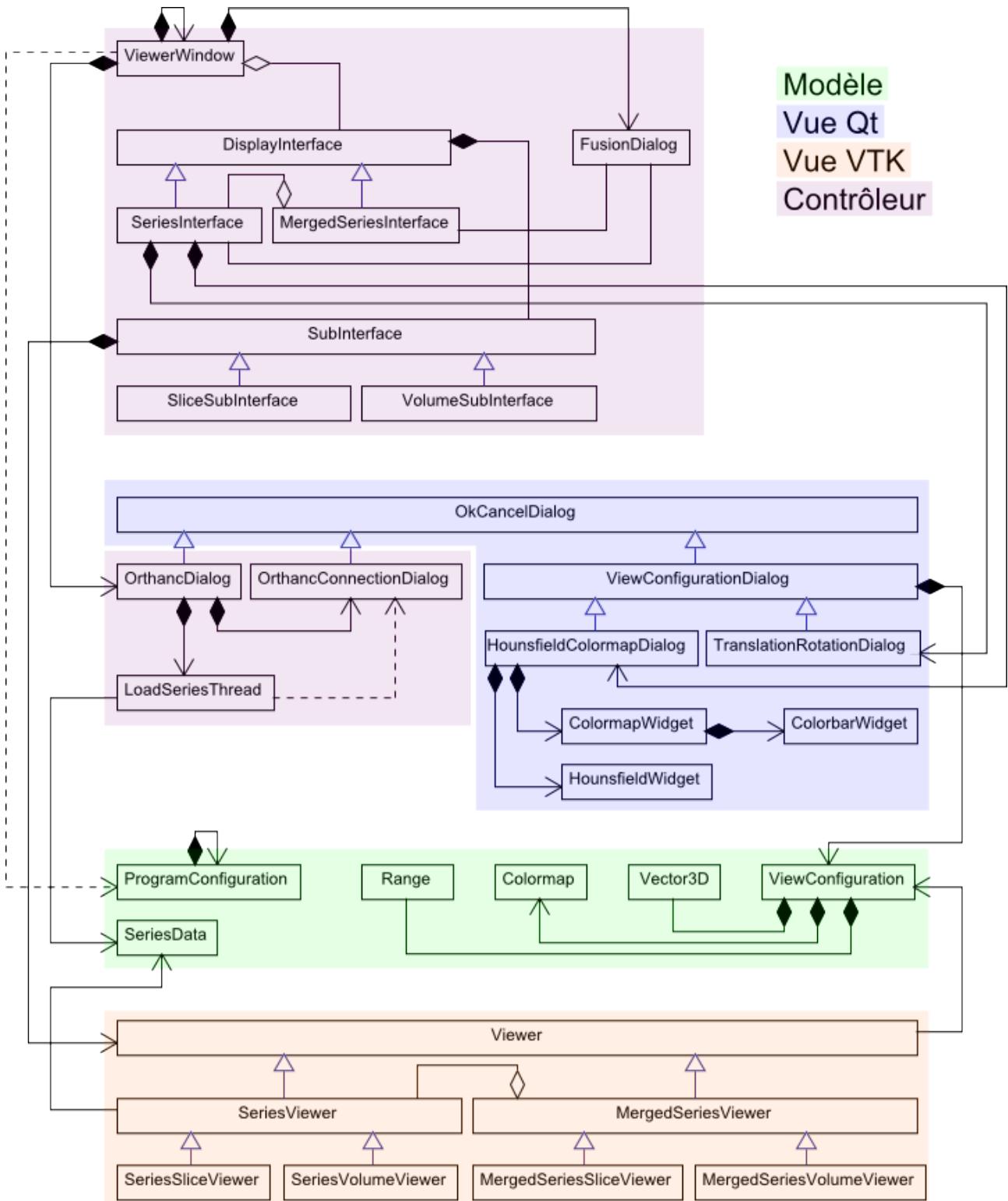


FIGURE 5.7 – Diagramme de classe final

# Chapitre 6

## Résultats

Ce chapitre présentera une série de captures d'écran du logiciel en état de fonctionnement. Chaque fonctionnalité décrite dans le chapitre précédent sera illustrée.

### 6.1 Lancement du logiciel

La figure 6.1 représente l'image d'ouverture pour le chargement du logiciel.



FIGURE 6.1 – Splash screen Orthanc

La figure 6.2 représente la boîte de dialogue de connexion au serveur Orthanc et l'erreur qui survient dans le cas où la connexion est impossible.



FIGURE 6.2 – Connexion à Orthanc

## 6.2 Chargement d'une série DICOM

Une fois le logiciel lancé, il propose automatiquement de charger une série à l'aide de la boîte de dialogue présentée sur la figure 6.3. On peut voir aussi que, pendant le chargement, une barre de progression permet de suivre l'avancement du chargement.

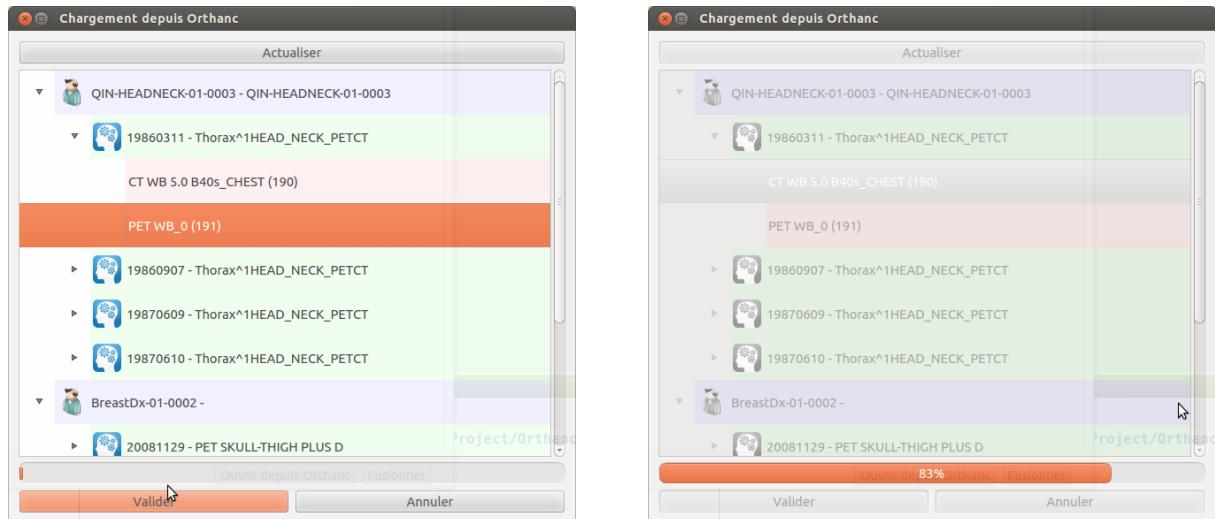


FIGURE 6.3 – Navigation dans Orthanc

Une fois une série chargée, l'interface de visualisation est celle présentée sur les figures 6.4 et 6.5.

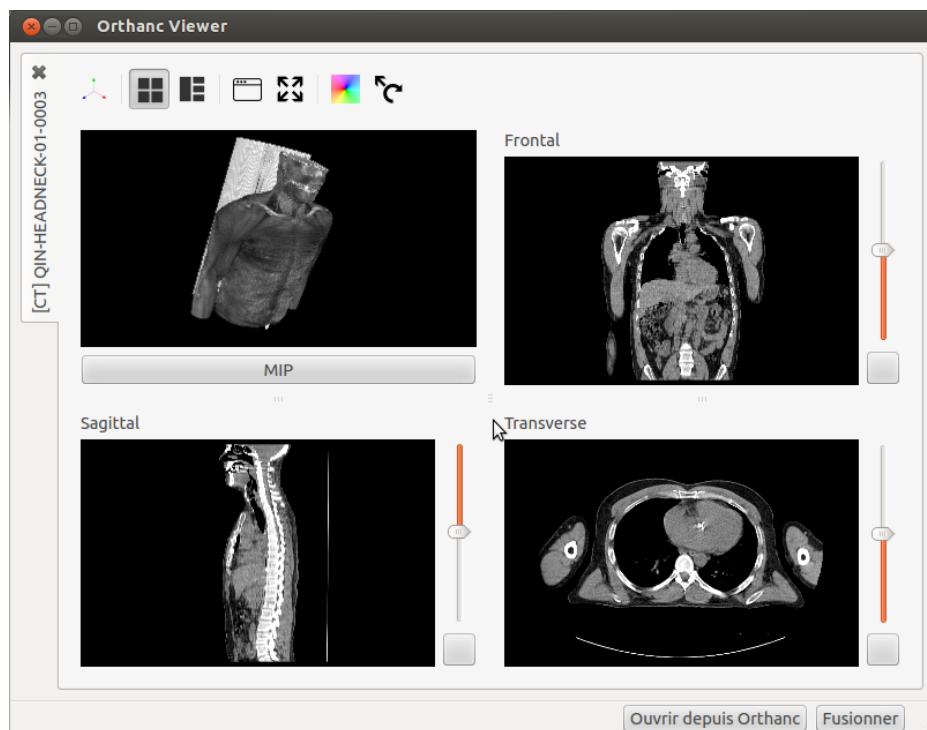


FIGURE 6.4 – Visualisation de base d'une série CT

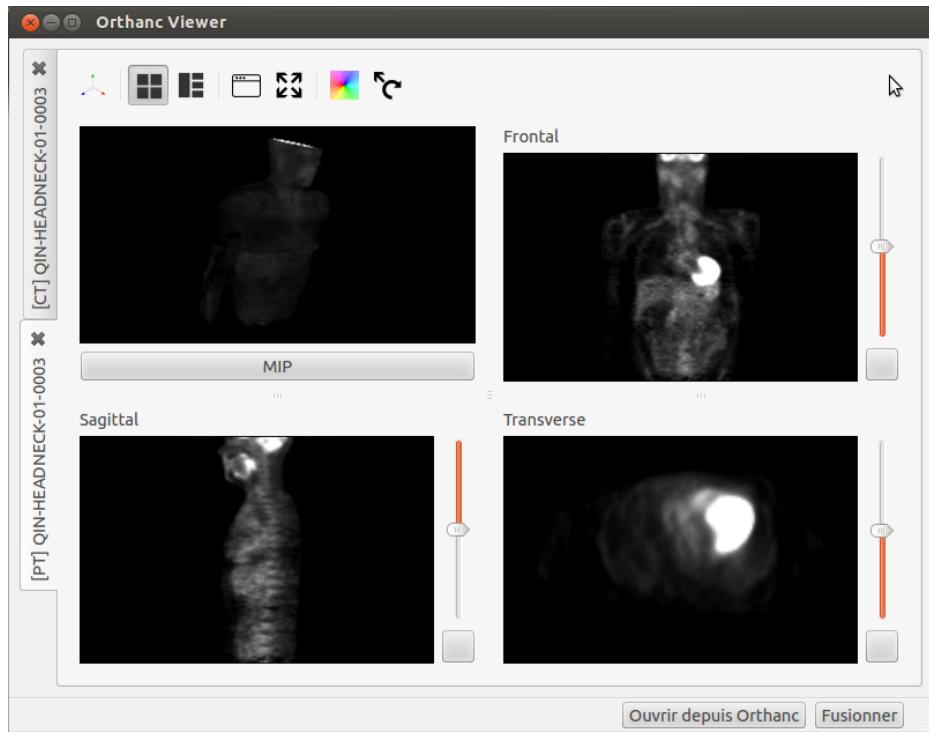


FIGURE 6.5 – Visualisation de base d'une série PET

Les vues permettent aussi de visualiser les axes orthonormés de la scène, ce qui peut être utile au moment d'effectuer une translation ou une rotation. En appuyant sur le bouton prévu à cet effet, on arrive donc au résultat présenté sur la figure 6.6.

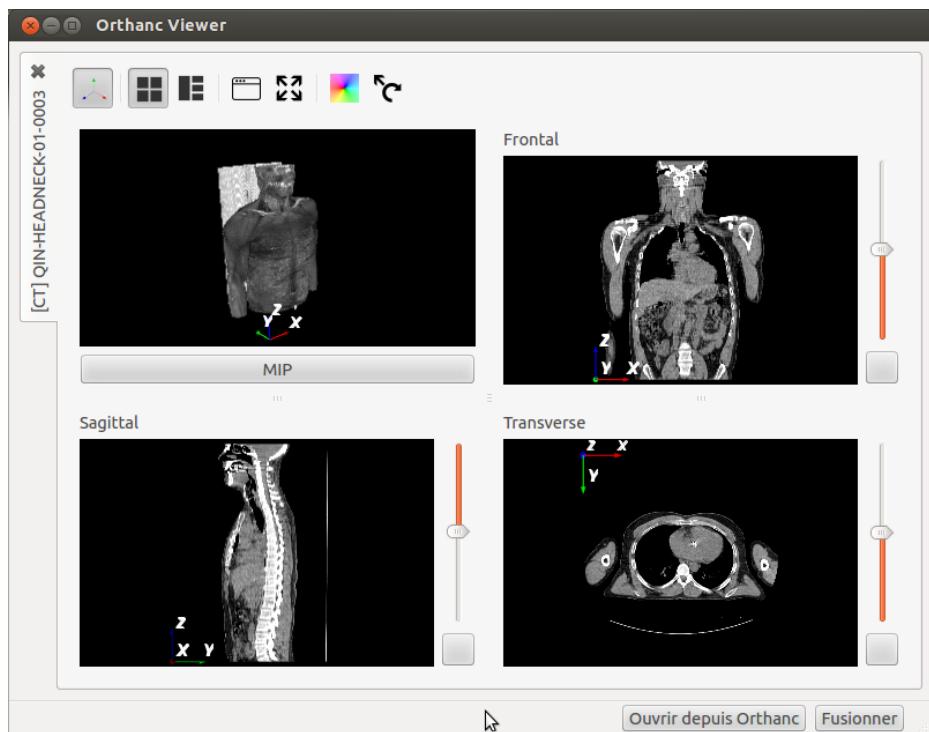


FIGURE 6.6 – Visualisation de base avec affichage des axes à l'intérieur des vues

### 6.3 Application de transformation d'image

La figure 6.7 présente le résultat d'un déplacement du slider prévu pour passer d'une coupe à une autre.

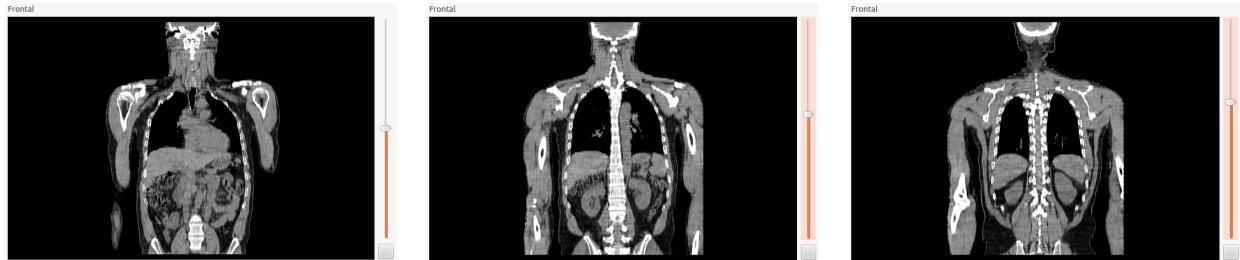


FIGURE 6.7 – Déplacement à travers les coupes à l'aide du « slider »

Pour paramétriser le fenêtrage et la coloration des éléments visualisés, un simple clic sur le bouton approprié ouvre la boîte de dialogue présentée sur la figure 6.8. Cette figure présente aussi la boîte de dialogue avec un preset choisi et une colormap appliquée. Notons que les boîtes de dialogue étant transparentes, les vues sont visibles à travers.

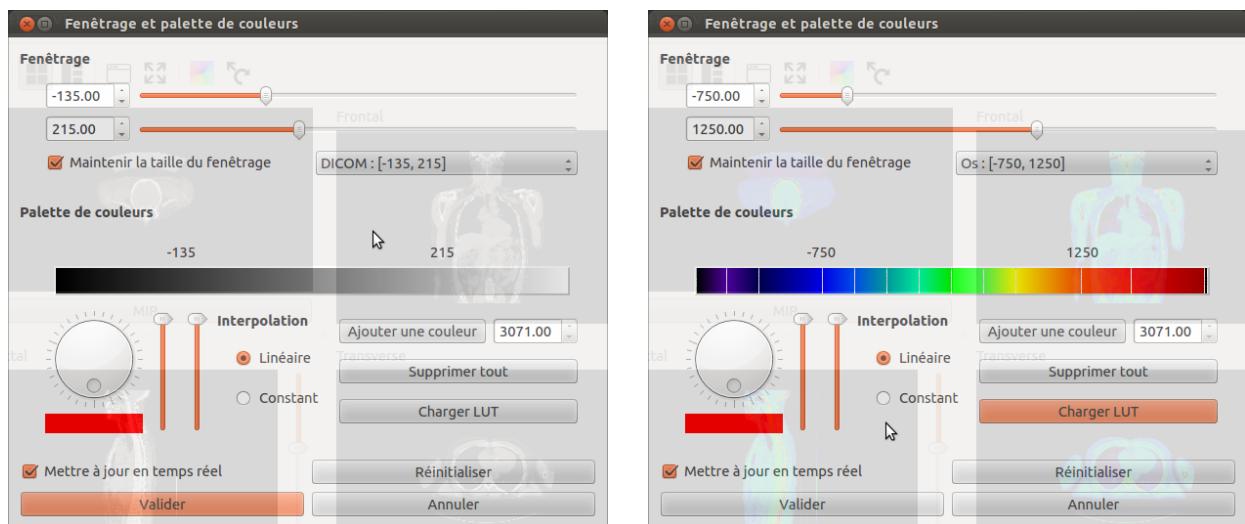


FIGURE 6.8 – Boîte de dialogue pour le contrôle du fenêtrage et des couleurs

La figure 6.9 présente une comparaison de différents fenêtrages utilisés sur des images et volumes CT. Notons que le résultat est aussi présenté pour le volume avec le mode MIP activé.

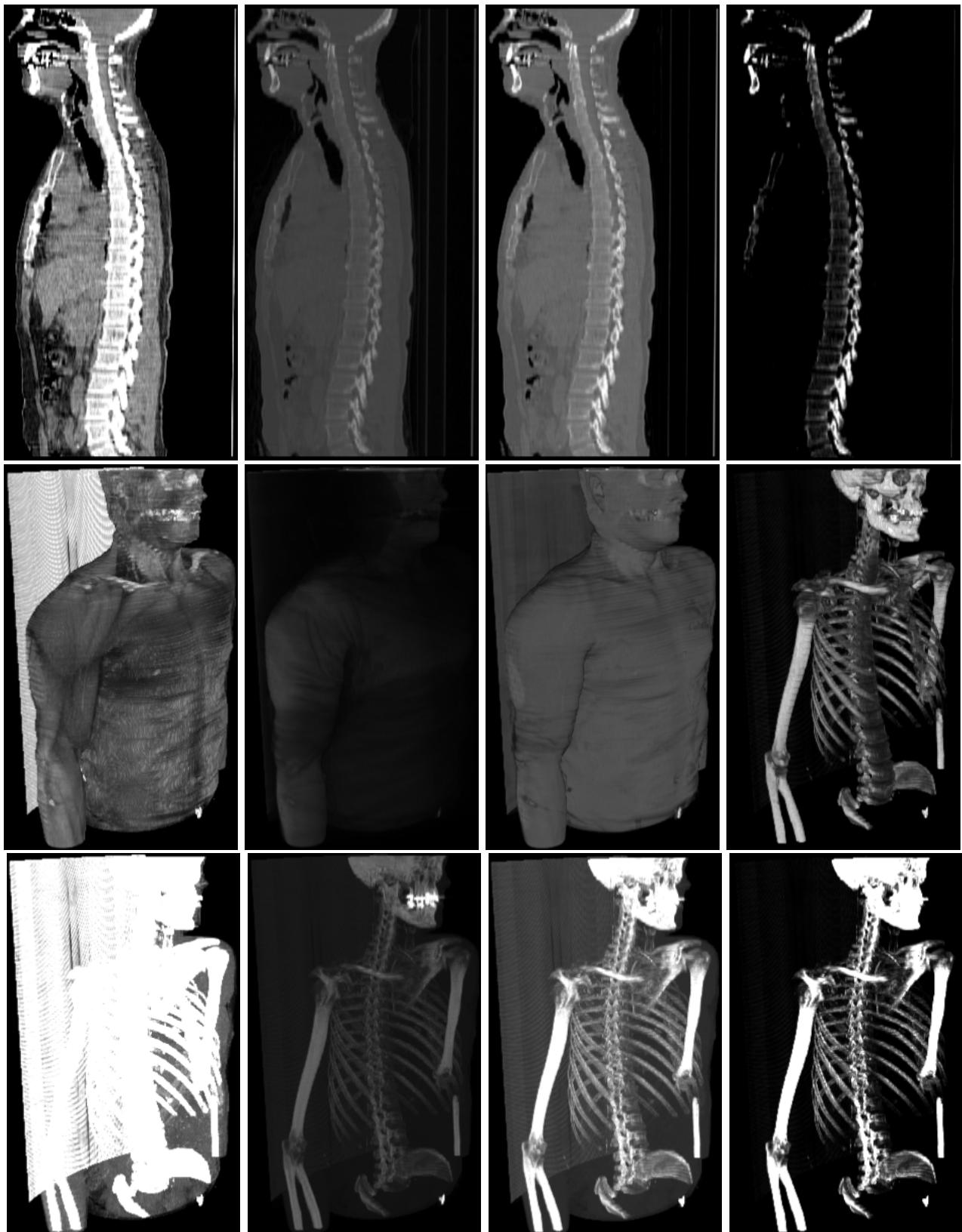


FIGURE 6.9 – Fenêtrage Hounsfield - De haut en bas : slice, volume et volume MIP - De gauche à droite : fenêtrage de base (spécifié par la série DICOM), fenêtrage maximum (base pour VTK), fenêtrage adapté aux os, fenêtrage "squelette"

Sur la figure 6.10, une comparaison de l'affichage d'un volume PET lorsque le mode MIP est activé ou non est présentée. Notons, pour rappel, que la figure 6.9 présentée précédemment montrait déjà une comparaison utile pour mieux comprendre le mode MIP.

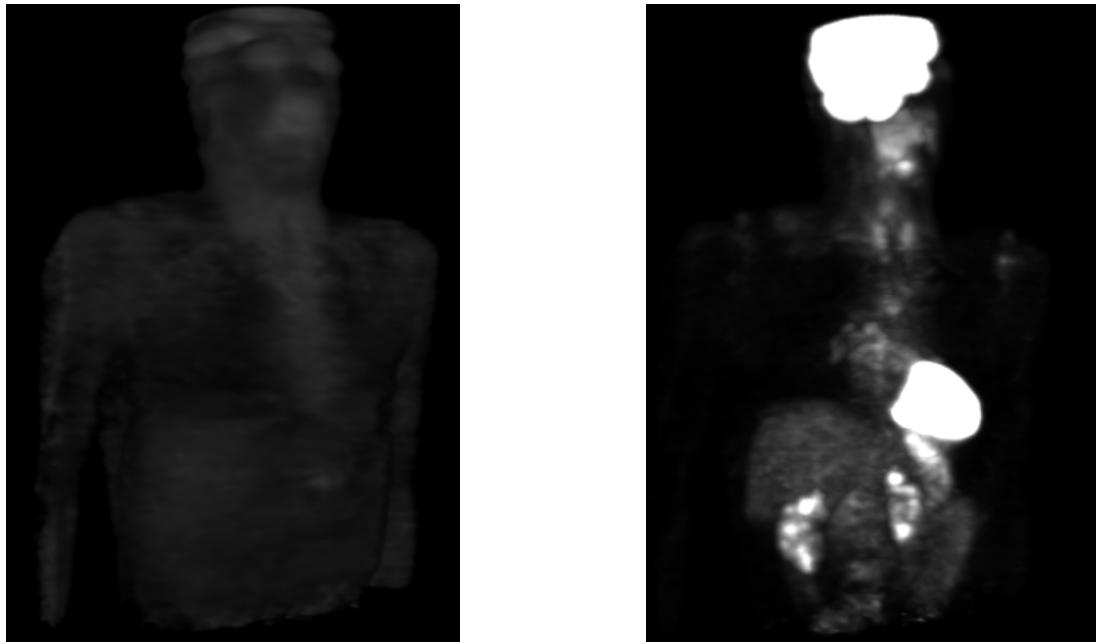


FIGURE 6.10 – A gauche, un volume PET et à droite, l'affichage MIP de ce même volume

La figure 6.11 présente le résultat d'une coloration personnalisée aux tons bleutés ajoutés à l'aide de la même boîte de dialogue que pour le fenêtrage.

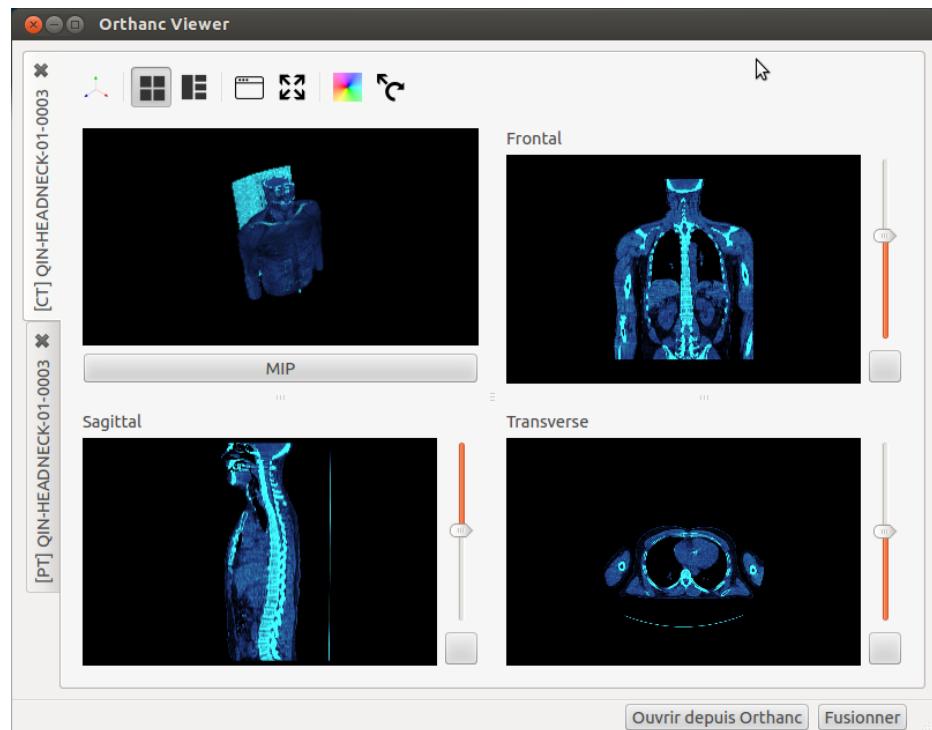


FIGURE 6.11 – Ajout d'une coloration personnalisée

Une coloration à l'aide d'un fichier LUT nommé « HOT » est utilisée et présentée sur la figure 6.12. Celle-ci utilise un dégradé allant du noir au blanc en passant par le rouge et le jaune.

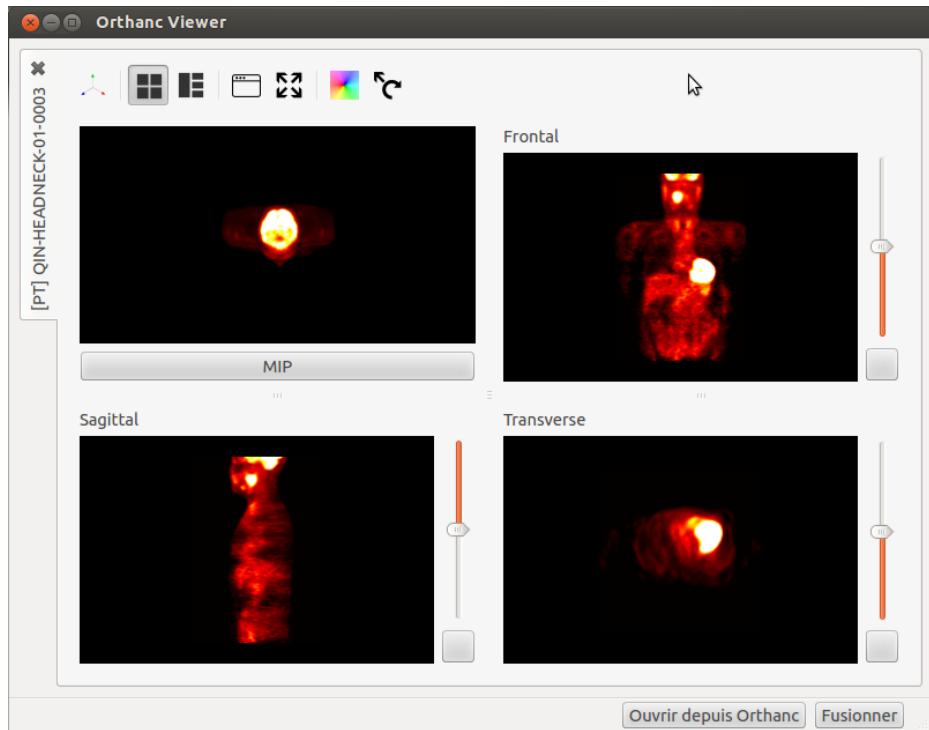


FIGURE 6.12 – Coloration à l'aide d'un fichier LUT « HOT »

Sur la figure 6.13, une comparaison de deux fenêtrages employés après application d'une « colormap » nommée « NIH » est mise en évidence. Comme on peut le voir, cette colormap utilise des tons noirs et violets pour les valeurs faibles d'intensité, des tons bleus, verts et jaunes pour les valeurs moyennes et des tons rouges et noirs pour les valeurs élevées. Le noir présente donc les extrêmes.

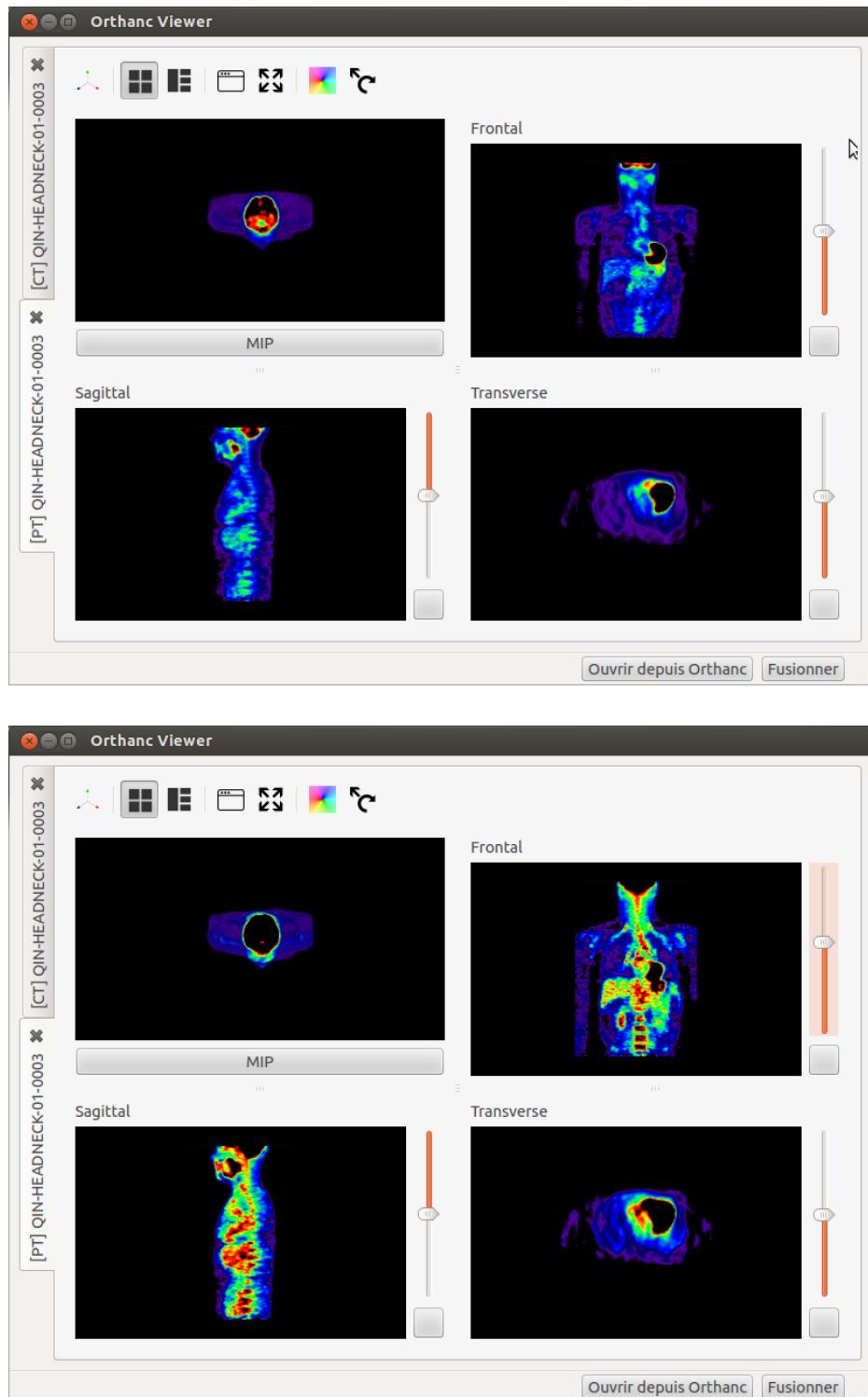


FIGURE 6.13 – Coloration à l'aide d'un fichier LUT « NIH » pour différentes valeurs de fenêtrage

## 6.4 Modification du layout de visualisation

Cette section présente la flexibilité du logiciel vis-à-vis de son interface. Sur la figure 6.14, on peut voir de quelle façon les vues peuvent être redimensionnées à l'intérieur de la fenêtre.

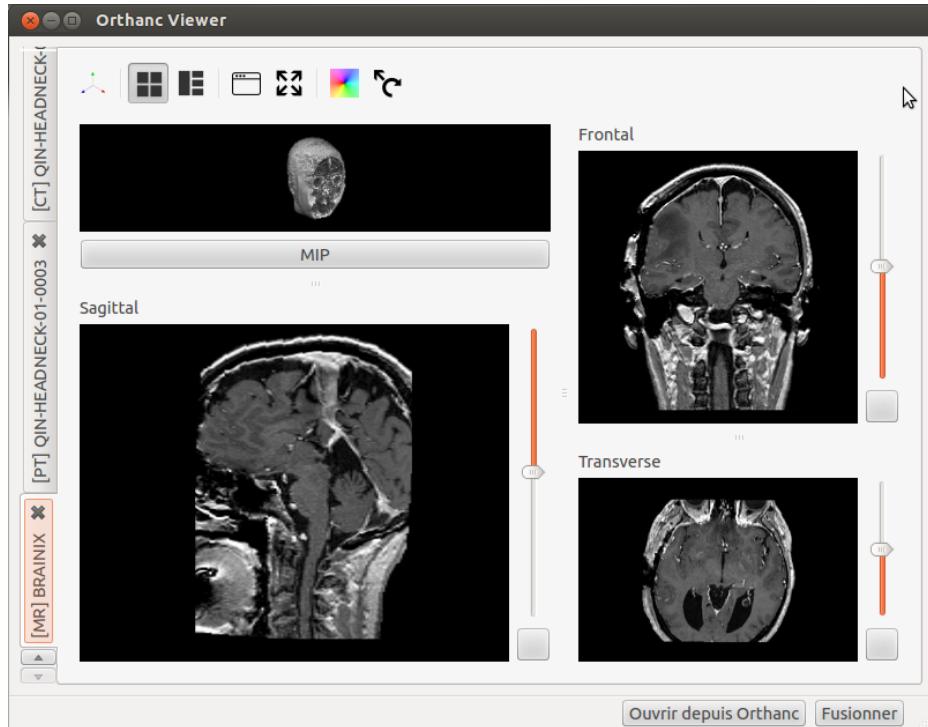


FIGURE 6.14 – Redimensionnement customisé des vues

La figure 6.15 présente (pour deux vues) la possibilité de masquer trois des quatre vues pour afficher la quatrième sur tout l'espace de la fenêtre. Pour activer ou désactiver ce mode, il suffit d'un simple double clic sur la vue concernée.

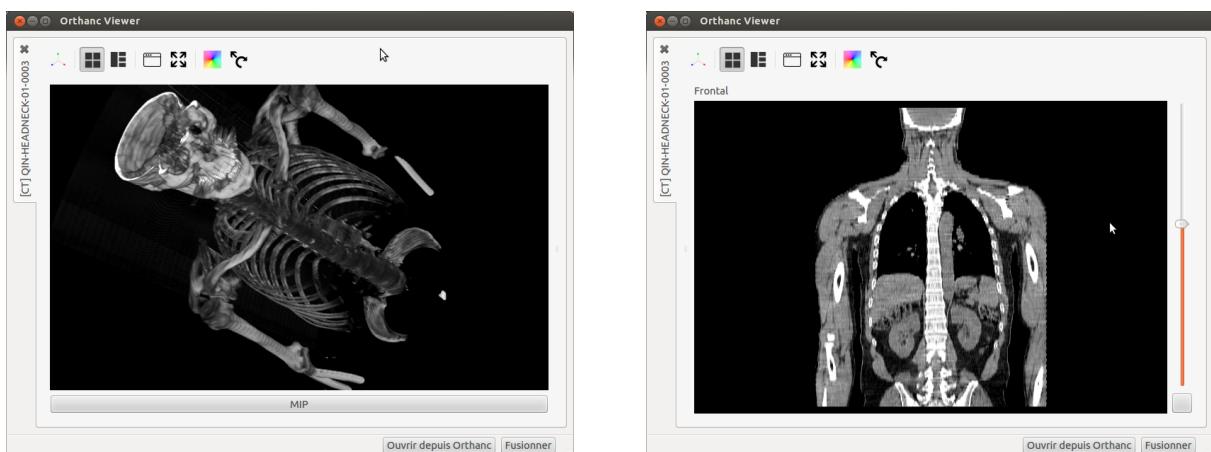


FIGURE 6.15 – Affichage d'une vue unique sur l'intégralité de la fenêtre

Comme précédemment expliqué, il est aussi possible de mettre l'interface d'une série en plein écran grâce à un simple clic sur le bouton correspondant. Cette possibilité est présentée sur la figure 6.16. Comme on peut le voir, ceci permet de masquer les onglets de série et la

barre d'état. Notons que, évidemment, il est possible de mettre une vue (et non une série) en plein écran en combinant le mode plein écran et un double clic sur la vue désirée (comme vu sur la figure 6.15).

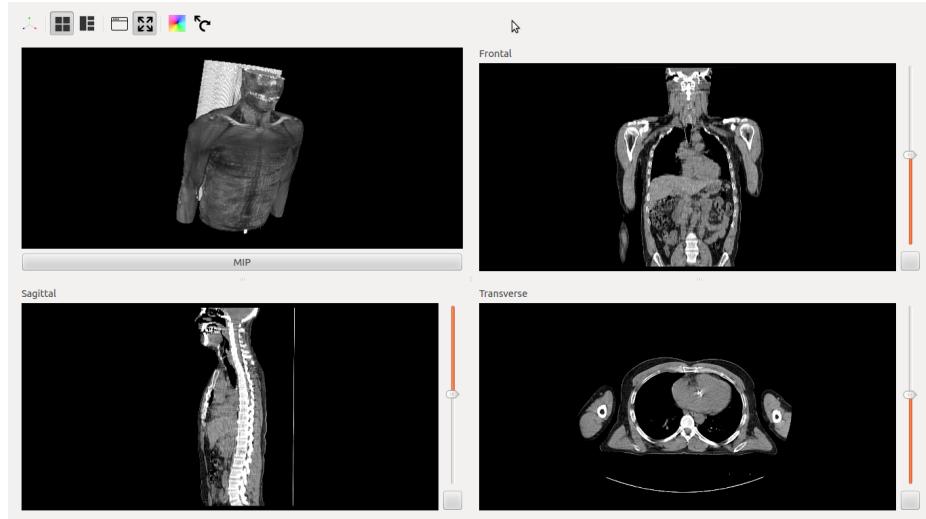


FIGURE 6.16 – Affichage d'une série en plein écran

Lors de l'enquête auprès du milieu hospitalier, deux layouts de visualisation ont été proposés et, au final, il a été décidé de permettre l'utilisation des deux modes. La figure 6.17 présente les deux modes de visualisation, entre lesquels on peut naviguer en choisissant le bouton approprié.

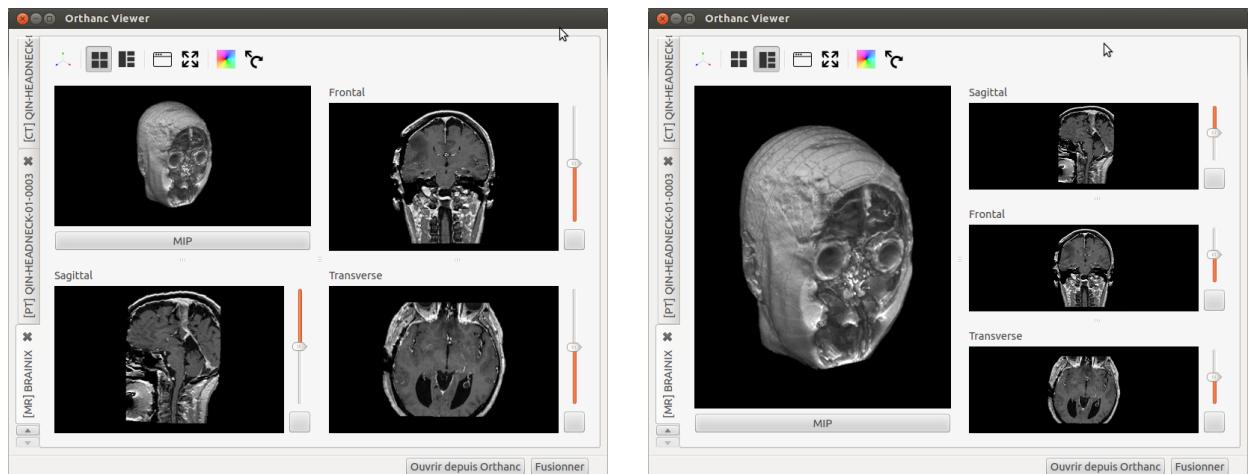


FIGURE 6.17 – Agencement des vues selon les deux layouts

La dernière possibilité offerte de l'interface est le déplacement d'une série vers une fenêtre indépendante, comme l'illustre la figure 6.18. L'introduire à nouveau dans le système d'onglet est bien entendu tout aussi simple.

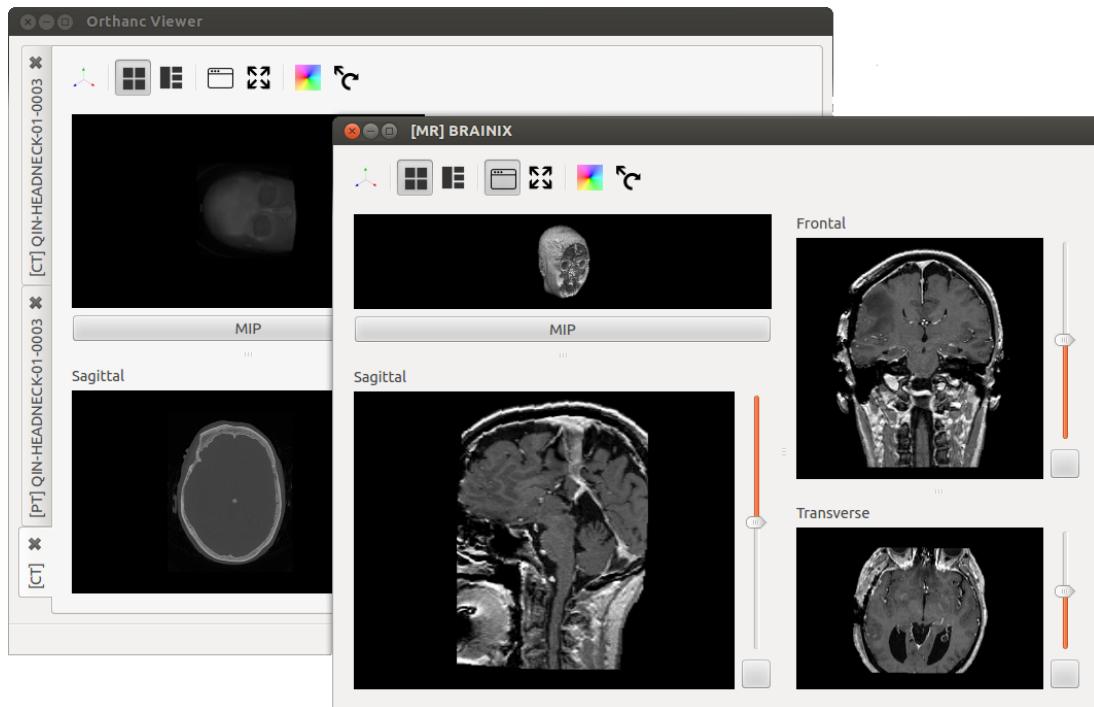


FIGURE 6.18 – Détachement d'une série vers une fenêtre indépendante

## 6.5 Fusion des séries

À l'aide de la boîte de dialogue présentée sur la figure 6.19, une fusion peut être effectuée. Notons que si aucune fusion n'a encore été créée, l'invitation à en créer une est automatiquement proposée à l'ouverture de cette fenêtre.

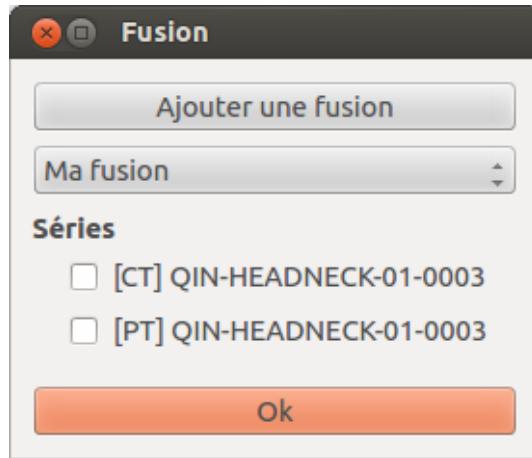


FIGURE 6.19 – Interface de sélection des séries à fusionner

Sur la figure 6.21, on peut observer le résultat juste après fusion d'une série CT et d'une série PET utilisant une coloration « NIH ». Dans la barre d'outil, on peut voir la possibilité de sélectionner la série que l'on veut personnaliser. On peut modifier l'opacité de la série sélectionnée à l'aide du slider à côté. Pour rappel, par défaut, la première série à entrer dans l'interface de fusion se voit attribuer une opacité de 1 tandis que les suivantes seront de 0.5. Aussi, on peut

toujours modifier le fenêtrage et la coloration de la série sélectionnée.

Comme on peut le voir, la fusion n'est pas complète car les séries CT et PET ne sont pas alignées. Dès lors, à l'aide de la boîte de dialogue visualisée sur la figure 6.20, on peut effectuer une translation pour parvenir à cet alignement.

Sur la figure 6.22, on peut observer le résultat de l'alignement. Notons que la série qui a été déplacée est la série PET. Ceci explique qu'à positions égales des sliders contrôlant le déplacement à travers les coupes, les coupes PET affichées sont différentes tandis que les coupes CT n'ont pas été modifiées.

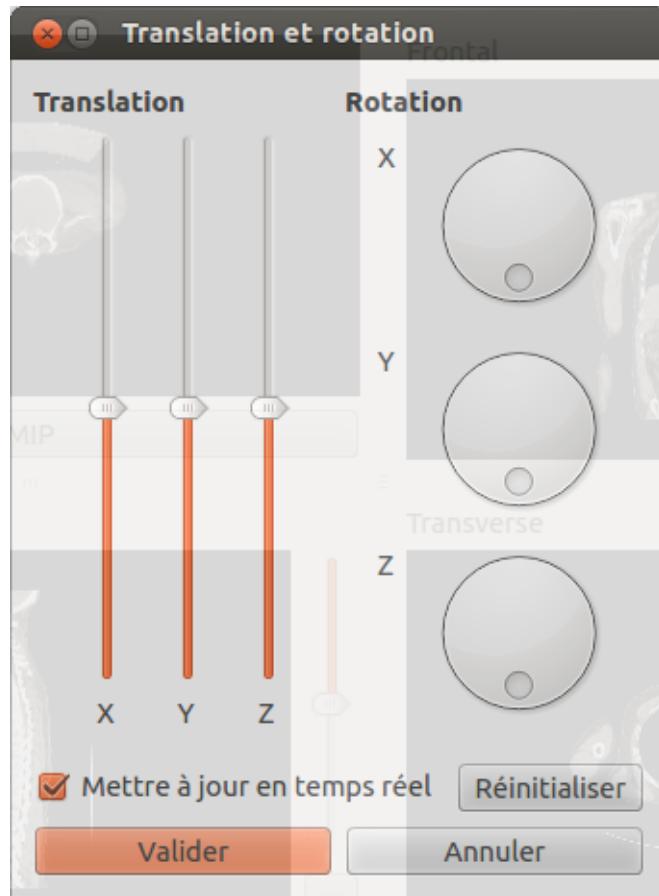


FIGURE 6.20 – Boîte de dialogue pour le recalage manuel des séries fusionnées

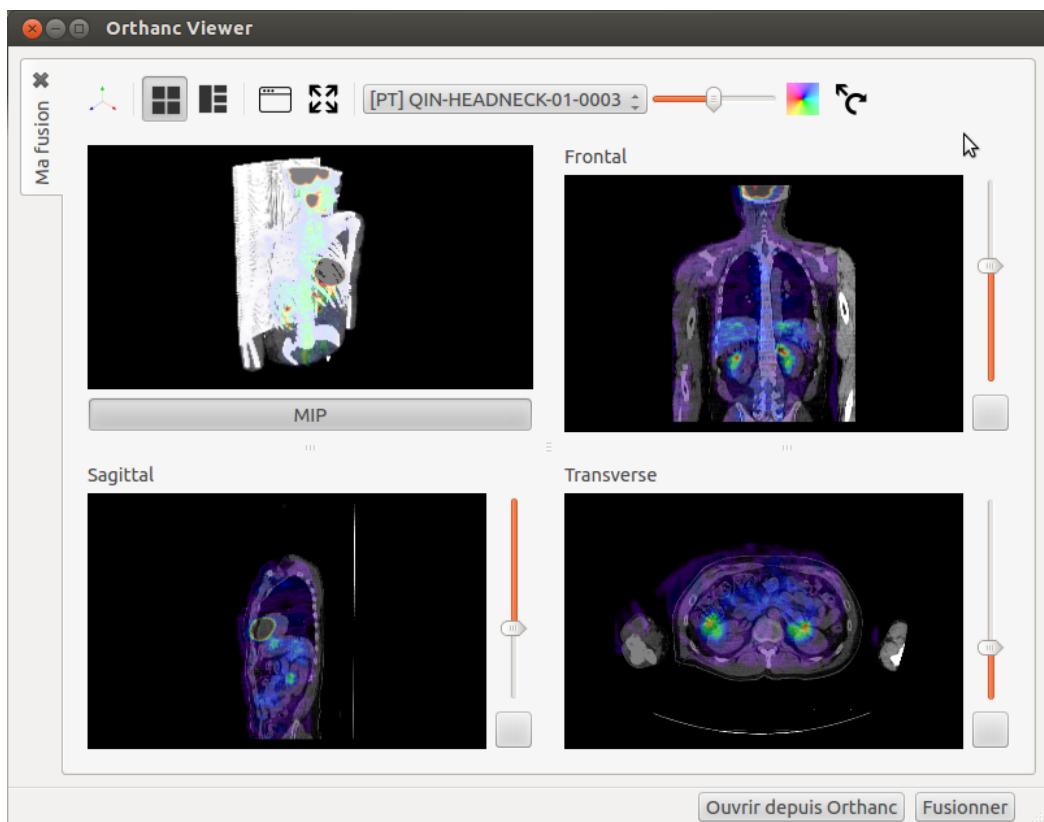


FIGURE 6.21 – Fusion d'une série CT avec une série PET colorée, avant recalage

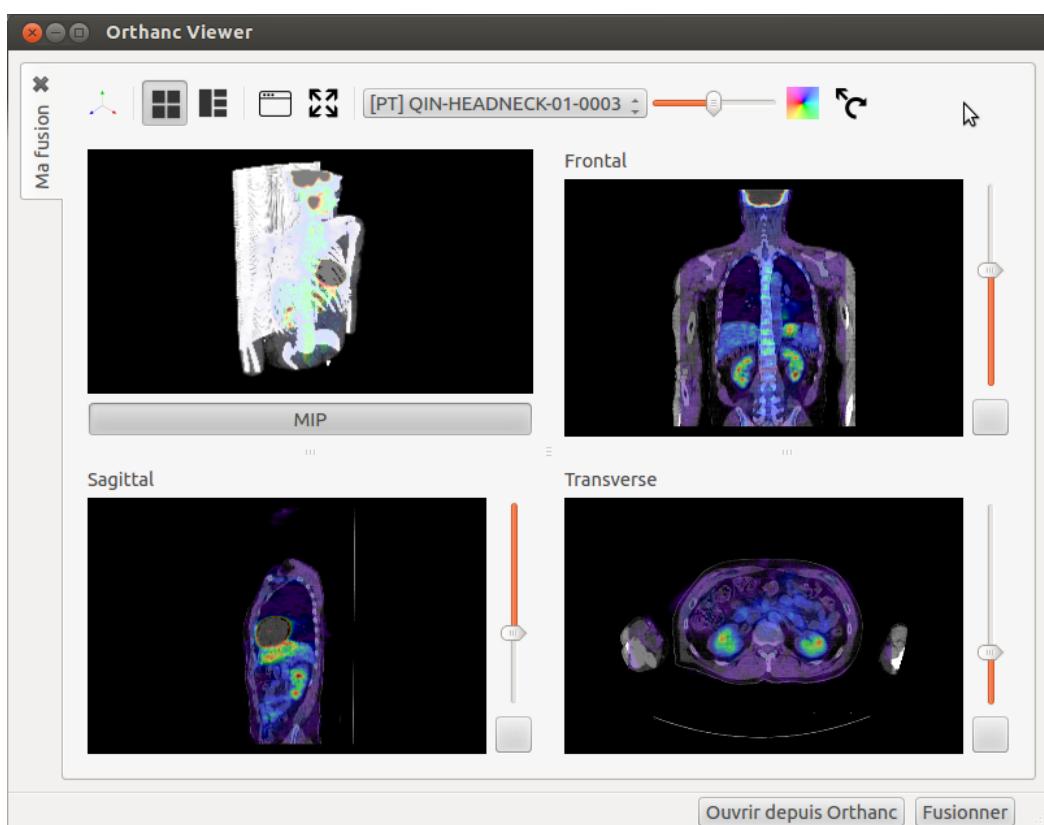


FIGURE 6.22 – Fusion d'une série CT avec une série PET colorée, après recalage

# Chapitre 7

## Améliorations futures

La conception du logiciel s'inscrit dans un travail plus large et il est évident que celui-ci verra son développement continuer d'une manière ou d'une autre. Ce chapitre regroupe quelques éléments n'ayant pas été réalisés lors de ce TFE et qui amélioreront plus tard les fonctionnalités du logiciel.

### 7.1 Visualisation d'autres modalités 3D

Le logiciel est pour l'instant adapté à la visualisation d'imagerie CT et PET ou d'autres formes d'imageries médicales. Certaines modalités nécessitent cependant une nouvelle phase de développement. C'est notamment le cas pour le format RT-STRUCT ou les fichiers vidéos 2D+t et 3D+t.

Ces améliorations ont cependant besoin d'une évolution de la bibliothèque Orthanc.

### 7.2 Mesures

Pour l'instant, les visionneuses (coupes et volumes 3D) affichent uniquement les données transformées. Il serait judicieux de pouvoir prendre des mesures de longueur sur l'image ou afficher d'autres informations utiles.

### 7.3 Annotations

Dans l'état actuel du logiciel, l'utilisateur n'a aucune possibilité de « sauvegarde ». Il pourrait y avoir besoin de commenter les images ou tout simplement de retenir le fenêtrage et/ou colormap utilisés pour une utilisation ultérieure. La division du logiciel selon une architecture MVC comme présenté précédemment permettra de sérialiser plus facilement les données « sauvegardables » pour les charger par après sur une autre instance du logiciel. Cette architecture MVC a donc déjà été un premier pas vers d'autres améliorations.

## 7.4 Alignement automatique des séries

Actuellement, le logiciel permet de fusionner (superposer) différentes séries et de les aligner manuellement. La fonctionnalité future évidente serait un travail d'alignement automatique.

Cette phase d'automatisation fait d'ailleurs actuellement l'objet d'un travail de doctorat et celui-ci pourrait être intégré au logiciel.

## 7.5 Autres

D'une manière générale, le logiciel est conçu pour permettre son évolution. L'ergonomie peut toujours être améliorée. On peut aussi penser à la convention médicale au niveau des orientations des vues. Celles-ci pourraient être décidées par l'utilisateur, d'autant plus que certains fichiers DICOM nécessitent cette possibilité pour être affichés comme il le faudrait. On peut aussi penser à une interpolation entre coupes qui permettrait d'améliorer la précision d'une fusion où le nombre de coupes des différentes séries n'est pas identique.

# Chapitre 8

## Conclusions

Comme cela a été montré, l'imagerie médicale est un sujet vaste et la visualisation de fichiers DICOM est une tâche ardue tant le standard est complexe. Ce TFE a permis d'en fixer les parties les plus importantes, à savoir :

- Naviguer parmi l'ensemble des patients et de leurs examens intuitivement à l'aide d'Orthanc
- Visualiser le volume 3D et ses coupes selon les plans frontaux, sagittaux et transverses
- À l'aide de commandes intuitives, se déplacer à travers les données, les colorer et atteindre une visualisation médicale utile en quelques clics
- Permettre une utilisation intuitive et personnalisable de l'interface
- Superposer toutes formes d'imageries, et permettre leur alignement manuel

Au-delà de l'ergonomie du logiciel, le code a lui aussi été travaillé dans l'optique d'un développement futur. Sa documentation et son organisation (MVC) devra permettre sa distribution et son utilisation en milieu médical.

## Informations sur le code final

Le code final contient 9647 lignes de code (293337 caractères) pour 118 fichiers. Parmi celles-ci, 3208 lignes sont consacrées à la documentation doxygen. La répartition est la suivante :

- Les fichiers `main.cpp` et `main.h` : 79 lignes
- Les 12 fichiers de modèle : 2003 lignes
- Les 16 fichiers de vue Qt : 2121 lignes
- Les 14 fichiers de vue VTK : 1636 lignes
- Les 22 fichiers de contrôleur : 3094 lignes
- Les 52 fichiers de l'espace de nom `customwidget` : 714 lignes

Rappelons que, notamment, la frontière entre la vue Qt et le contrôleur est en partie arbitraire et que donc, cette répartition est présentée à titre indicatif.

# Références bibliographiques

- [1] Mathieu Nebra, Matthieu Schaller, “Introduction à Qt.” <http://fr.openclassrooms.com/informatique/cours/programmez-avec-le-langage-c/introduction-a-qt>. Consultation : 2013-2014.
- [2] “Documentation | Qt Project.” <http://qt-project.org/doc/>. Consultation : 2013-2014.
- [3] “VTK - The Visualization Toolkit.” <http://www.vtk.org/VTK/help/documentation.html>. Consultation : 2013-2014.
- [4] Franck Hecht, “Initiation à Doxygen pour C et C++.” <http://franckh.developpez.com/tutoriels/outils/doxygen/>, Septembre 2007. Consultation : 2013.
- [5] National Electrical Manufacturers Association, “Digital Imaging and Communications in Medicine (DICOM) - Part 3 : Information Object Definitions,” 2009.
- [6] “DICOM Tags List.” <http://www.dicomtags.com/>. Consultation : 2013-2014.
- [7] Bruno Van Oystaeyen, “Fenêtrage.” <http://www.phys4med.be/construction/fenetrage>, Février 2012. Consultation : 2014.
- [8] “Hounsfield (échelle) — Wikipédia.” [http://fr.wikipedia.org/wiki/Hounsfield\\_%28%C3%A9chelle%29](http://fr.wikipedia.org/wiki/Hounsfield_%28%C3%A9chelle%29). Consultation : 2013-2014.
- [9] Bernard Boigelot, “Object-oriented software engineering.” <http://www.montefiore.ulg.ac.be/~boigelot/cours/se/slides/se.pdf>, Année scolaire 2013-2014. Consultation : 2013-2014.

# Annexe A

## Formulaire d'enquête

### A.1 Formulaire

#### Logiciel d'imagerie médicale

Répondre à ce formulaire permettra d'aider à la réalisation d'un logiciel d'imagerie médicale léger, pensé pour l'utilisation en milieu hospitalier.

Ce logiciel est développé dans le cadre d'un travail de fin d'étude, utilisant Orthanc.

Merci d'avance pour votre collaboration.

\*Obligatoire

#### Informations personnelles

##### Voulez-vous décliner votre identité?

Ceci n'est évidemment pas obligatoire mais peut aider à gérer les réponses du formulaire

##### Quelle profession exercez-vous? \*

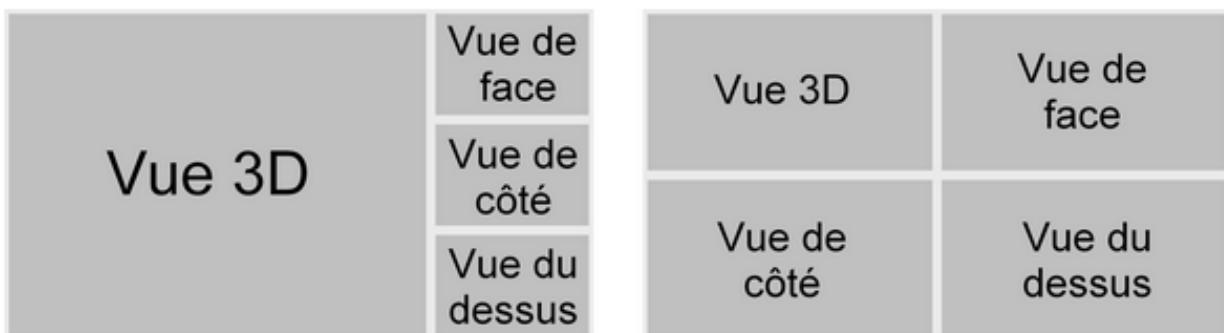
Connaître votre fonction au sein de l'hôpital aidera à se faire une idée des désirs ressortant des personnes occupant des fonctions similaires

- Radiothérapeute
- Physicien médical
- Autre :

## Ergonomie et design

Cette partie vise à connaitre vos desiderata au niveau de l'ergonomie et du design du logiciel

## Interface de visualisation



Indiquez dans quelle mesure vous trouvez les possibilités suivantes utiles \*

Certains points se réfèrent aux images présentées ci-dessus.

## Les fonctionnalités

Cette partie vous permet de préciser l'utilité de certaines fonctionnalités dont pourrait disposer le programme.

**Indiquez dans quelle mesure vous trouvez les fonctionnalités suivantes utiles \***

## Remarques complémentaires

### D'autres idées ou remarques?

Faites part de toute information pouvant aider au projet ! Le moindre avis sur ce qui peut être utile ou inutile dans un logiciel d'imagerie médicale est le bienvenu.

**Envoyer**

N'envoyez jamais de mots de passe via Google Formulaires.

**100 % : vous avez réussi.**

## A.2 Résultats

